The Thesis Committee for Joshua Murry Williams
Certifies that this is the approved version of the following thesis:

Improved Manipulator Configurations for Grasping and Task Completion Based on Manipulability

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor: _____
Sheldon Landsberger

Co-Supervisor: _____
Mitchell Pryor

Improved Manipulator Configurations for Grasping and Task Completion Based on Manipulability

by

Joshua Murry Williams, B.A.

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

December 2010

Dedication

To my parents

Acknowledgements

I would like to thank Dr. Mitchell Pryor for his guidance during the research process and for helping me become a better researcher. I would like to thank Dr. Sheldon Landsberger for his encouragement and direction of our program. I cannot forget to thank all my family and friends who have prayed for and encouraged me throughout my graduate work. Thanks be to God for providing me the opportunity for graduate studies and for sustaining me.

This work was partially funded by the National Nuclear Security Administration under the University Research Program in Robotics, grant #DE-FG52-06NA25591 and by Los Alamos National Labs under project #79506-001-10.

December 3, 2010

Abstract

Improved Manipulator Configurations for Grasping and Task Completion Based on Manipulability

Joshua Murry Williams, M.S.E.

The University of Texas at Austin, 2010

Supervisor: Sheldon Landsberger

Co-Supervisor: Mitchell Pryor

When a robotic system executes a task, there are a number of responsibilities that belong to either the operator and/or the robot. A more autonomous system has more responsibilities in the completion of a task and must possess the decision making skills necessary to adequately deal with these responsibilities. The system must also handle environmental constraints that limit the region of operability and complicate the execution of tasks. There are decisions about the robot's internal configuration and how the manipulator should move through space, avoid obstacles, and grasp objects. These motions usually have limits and performance requirements associated with them.

Successful completion of tasks in a given environment is aided by knowledge of the robot's capabilities in its workspace. This not only indicates if a task is possible but

can suggest how a task should be completed. In this work, we develop a grasping strategy for selecting and attaining grasp configurations for flexible tasks in environments containing obstacles. This is done by sampling for valid grasping configurations at locations throughout the workspace to generate a *task plane*. Locations in the task plane that contain more valid configurations are stipulated to have higher dexterity and thus provide greater manipulability of targets. For valid configurations found in the plane, we develop a strategy for selecting which configurations to choose when grasping and/or placing an object at a given location in the workspace.

These workspace task planes can also be utilized as a design tool to configure the system around the manipulator's capabilities. We determine the quality of manipulator positioning in the workspace based on manipulability and locate the best location of targets for manipulation. The knowledge of valid manipulator configurations throughout the workspace can be used to extend the application of task planes to motion planning between grasping configurations. This guides the end-effector through more dexterous workspace regions and to configurations that move the arm away from obstacles.

The task plane technique employed here accurately captures a manipulator's capabilities. Initial tests for exploiting these capabilities for system design and operation were successful, thus demonstrating this method as a viable starting point for incrementally increasing system autonomy.

Table of Contents

List of Tables

List of Figures

Chapter 1 **Introduction**

Robots perform a wide variety of useful tasks that commonly involve repetition, high levels of precision, heavy lifting, etc. These tasks require maneuvers which are tedious, difficult, or impossible for humans to perform on their own. If the work occurs in an environment that is hazardous to humans, the use of robotics is further justified. Together, task and environmental constraints help drive the selection of robotic systems for particular applications.

Selecting the best type of system based on application constraints is a critical system design decision. Mobile systems are often deployed in unknown environments where tasks are not well-defined until the robot reaches the target area. Non-mobile systems, however, are often utilized where process tasks are well-defined and the environment surrounding the robot remains relatively static. This applies to robots used in manufacturing plants and in other industrial processes. Identical products and/or fixed handling are desired virtually 100% of the time and even small perturbations in the process may result in large and costly mistakes.

Improvements can be made by developing flexible non-mobile systems that have the ability to respond to changes, no longer constrained to static environments with predetermined and repetitive tasks. A more generalized system would be capable of supporting a wide variety of processes including variation in the sub-tasks of a given process. The system must also allow for the set of processes to be completed without knowing the schedule, order, or frequency that the processes will be completed. The robot could reside in a dynamic environment where the locations and numbers of targets and objects in its workspace could vary. Such an environment may even consist of a previously non-existent human presence either in the task space or the robot's operational

loop. As the tasks and environment become more complex, a robotic system needs more refined decision making.

1.1 ROBOTIC AUTONOMY

A system's level of decision making determines how much responsibility is placed on the robot and/or operator. Increasing the autonomy of a robotic system is tied to a reduction or elimination in the responsibility of an operator to direct or manage some aspect of the robot's execution of a task. These responsibilities are then taken care of by the control software, sensors, and hardware which decrease the need for human sensing and decision making. Researchers at Idaho National Lab (INL) have divided the autonomy spectrum into five discrete levels that begin with a teleoperation baseline. [Few, 2006]

Table 1-1: A five level breakdown of the autonomy spectrum [Few, 2006]

Autonomy Mode	Defines Task Goals	Supervises Direction	Motivates Motion	Prevents Collision
Teleoperation Mode	Operator	Operator	Operator	Operator
Safe Mode	Operator	Operator	Operator	Robot
Shared Mode	Operator	Operator	Robot	Robot
Collaborative Tasking Mode	Operator	Robot	Robot	Robot
Autonomous Mode	Robot	Robot	Robot	Robot

In order to incrementally move toward higher levels of autonomy, we must have some structure and understanding of what capabilities lead to greater autonomy. Therefore, it is useful to associate autonomous capabilities with well defined layers spanning a spectrum from pure teleoperation to full autonomy. Although the work of Few [2006] was directed primarily toward mobile manipulation, the general automation

goal of each step can be applied to fixed-base manipulators. [Kulkarni, 2008] Indeed, all the work discussed can generate tools for improving both robotic fields.

Our effort addresses *transitional levels of autonomy* for manipulation. In Table 1-2, we have defined ten levels from a baseline of tele-operation to an idealized autonomous system from a task autonomy perspective instead of INL's roles/responsibilities perspective. Each level provides capabilities to ease the control burden for the operator while still supporting the option to transition to lower or higher autonomy levels during operation if necessary.

Table 1-2: Transitional levels of autonomy [Pryor, 2010]

Level	Tele-op → Autonomy…
1	Reduce or eliminate operator's need to manage the robot's internal configuration.
2	Reduce or eliminate the operator's responsibility for avoiding undesired contact with the environment.
3	Reduce or eliminate the operator's responsibility for moving the robot to locations of interest.
4	Reduce or eliminate the operator's responsibility for selecting a proper grasping configuration for retrieving selected objects.
5	Allow the operator to quickly direct the system to complete tasks that involve subtasks completed as directed in levels 3 & 4 (such as pick & place).
6	Reduce or eliminate the operator's responsibility to avoid threshold forces for contact tasks such as opening a door or lifting items exceeding the system's payload.
7	Reduce or eliminate the levels of detail that are necessary for the operator to communicate a task (or subtasks) to the robotic system.
8	Integrate capability to complete task that require high levels of precision and/or the control a specific force profile
9	Reduce or eliminate the need for the operator to be in the loop for tasks that respond to non-operator, independent external events (i.e. timer on oven, low battery notification, etc.)
10	Based on prior tasks completed, the system anticipates future tasks to be completed based on historical use.

1.2 OBJECTIVES AND SCOPE

It will take the collaboration of numerous research areas to progress toward higher levels of autonomy. In regard to these levels, this research focuses on the selection of

manipulator configurations during grasping. Material grasping, moving, and placement is a large component in the execution of any larger task or process. In many application areas (i.e. manufacturing), materials are retrieved from the same known location every time, grasped using the same manipulator end-effector (EEF) orientation, moved along the same un-obstructed path, and placed in a predetermined location. Hard-coding of the robot's behavior is sufficient barring otherwise unexpected complications for these highly repetitive tasks, but it does not support on-demand flexibility. If modifications are necessary, the operator must shut down the system then provide some level of reprogramming, re-tooling, and/or workspace reconstruction.

Humans, on the other hand, have the ability to intuitively respond to changes in task and environment. We process information regarding what we see and then make decisions based on prior experiences and logic. In the simple task of moving an object, we gather information about the shape and weight of the object, what other objects/obstacles are around it, grasp the object with a favorable arm configuration, and then move the object. Commonly, the movements are over some path that is not exactly the same every time we repeat that process. This may occur because the initial position of an object may vary along with the final location and orientation depending on the state of the environment around the region of the workspace where we wish to move the object. **The objective of this research is to improve the decision making process that allows a robot to make these same strategic choices, thus, leading to a more autonomous system.**

This specific objective aligns with our research group's broader mission of attaining this flexibility and generalization using transitional levels of autonomy. The long-term goal is for the robot to make these decisions and minimize the role of humans

in the operational loop. To solve this problem, this effort addresses two high level objectives:

1. Accurately capture the kinematic capabilities of a robot in its workspace, and
2. Exploit our improved understanding of robotic capabilities and apply it to
    - System design/layout, and
    - System operation.

In our applications, we want to help the operator and system designer, who may be un-trained in robotics, make decisions about movements within the manipulator's workspace. These individuals commonly rely on intuition and trial-and-error to select kinematic configurations while performing tasks. This research specifically addresses choosing manipulator configurations for grasping before/after executing a motion trajectory. By including trajectory information in the analysis, we generate a more complex, but realistic understanding of a robotic system's capabilities.

In a grasping task, the manipulator must attain a *grasping configuration* to *grasp* (or hold) an object. This grasping configuration orients the gripper relative to the object being grasped, defines the gripper's internal configuration, and provides a set of joint angles for the arm. In this thesis, a *grasping strategy* is the process of determining how to select and achieve a grasping configuration.[1] We will focus on grasping strategies for fixed-base manipulators performing pick-and-place tasks, which is a common scenario in many processes. The relative simplicity of these types of tasks also makes them a good starting point for progressing toward more complex tasks. We will also look at conducting tasks in confined environments containing various obstacles with an emphasis

[1] In the literature, the phrase *grasping strategy* often refers to selecting the internal configuration of the gripper for grasping an object instead of the use here referring to selecting the manipulator's best configuration *for* grasping. The two areas of study are obviously tightly coupled. Selecting the best grasper configuration is an area of research that is also ongoing at The University of Texas at Austin.

on glovebox and other Department of Energy (DOE) related applications (see Section 1.3.3).

1.2.1 Contributions to the Autonomy Spectrum

To develop a grasping strategy that allows a system to grasp more autonomously, we must understand the areas of robotics linked with grasping tasks. These areas include task planning, layout analysis, modeling, and motion/trajectory planning. These subjects are reviewed more extensively in Chapter 3. Below is a brief discussion of the relevance of these fields to manipulator grasping:

- **Task Planning –** Many requirements and system constraints are derived from the task analysis, which divides the larger system processes into tasks and sub-tasks that represent low-level commands. The task analysis also determines the types of objects used and the grasps needed.
- **Layout Analysis** – If possible, the robot should execute tasks in dexterous regions of the workplace. This way, there is the option to choose between multiple grasping configurations for optimization purposes.
- **Modeling** – The system must know how objects can be grasped. At the same time, an accurate environmental model is needed to attain grasping configurations that avoid unwanted collisions.
- **Motion and Trajectory Planning** – Once the initial and final grasping configurations are determined, we need to know if a collision-free trajectory can be planned between them. We must also consider how the EEF motion through space affects manipulator dexterity.

In order to discuss how grasping influences the levels of autonomy, emphasis has been placed on describing the levels as they apply to task planning, layout analysis, modeling, and path planning. Below are some of the potential contributions and/or assumptions made at each automation level:

- **Level 1:** Optical sensing and path planning are performed by the operator but the inverse kinematics is taken care of by the control software, and joint limits, singularities, torque limits, etc, are automatically avoided.
- **Level 2:** The job of preventing collisions with the environment is no longer the sole responsibility of the operator. Information from the sensors and/or models allows for active detection and/or avoidance of collisions/obstacles. Motion is conducted between well-defined locations that are known to have a collision-free path between them.
- **Level 3:** Emphasis is placed on path planning and less direction is given from the operator concerning the frame locations that the manipulator traverses. The system acquires the ability to choose valid starting and ending locations and generates trajectories in an environment that may be dynamic.
- **Level 4:** Improved decision making for grasping and path planning is used to determine the optimum grasping configurations and trajectories for the execution of a task.
- **Level 5+:** The system autonomously takes care of the planning and selection of sub-tasks to execute the higher level task given.

Recent efforts (discussed in Section 3.1) have been performed for autonomy at Level 2 which assumed a known world model.[2] The current model is assumed to be static, so grasps, starting locations, and ending locations are pre-programmed into the system. These locations are selected based on the knowledge that simple trajectories can be calculated between the locations, as obstacles are not in the way.

This effort will develop and complete objectives for allowing functionality at Levels 3 and 4. The major difference at these levels is that it is not assumed that a given motion task is possible. The system must decide based on the supplied data whether it can find grasping configurations and paths that allow tasks to be executed more autonomously since the system no longer depends on the operator to make these decisions.

1.2.2 Supporting Technology

As previously mentioned, improved autonomy requires the integration of multiple technologies. Although the emphasis here is on manipulator grasping configurations and path planning, progression to higher levels of autonomy requires supporting technologies to address the additional issues associated with an increase in automation. These areas include world modeling, force/torque control, and operational software.

For a manipulator in a dynamic environment, the data obtained must produce an accurate world model for the robot to perform meaningful tasks. For this purpose, sensors are important for allowing the robot to behave intelligently within a dynamic environment. This is not limited to optical sensing, but also the ability to detect and measure forces/torques on the manipulator in order to create a more robust control scheme. This detection is also useful for contact tasks and steering the manipulator away

[2] Generating this model is part of other research currently underway at The University of Texas at Austin.

from hazards if force/torque limits are reached. Such capability can reduce harm to human operators if unwanted contact does occur. Systems must be capable of processing received data such that the robot can make intelligent task decisions. This data includes manipulator geometry, sensor data for world modeling, and input device signals.

1.3 MOTIVATING APPLICATIONS

The benefits of the proposed progression in the autonomy spectrum are motivated by needs in several application areas, including the need for safe interactions between robots and humans. Since this work integrates with other technologies and is also applicable to other areas not directly addressed, the combined advancements can have far reaching effects.

1.3.1 Human Augmentation

Constraints imposed by the workspace or operator can limit the ability to execute a desired task. The solution may be to scale back the supported tasks or to entirely redesign the workspace. In many cases, this is not feasible or may be unacceptable. Developing a system where robots work in conjunction with human operators can create a scenario where the worker's capabilities are extended. This is the idea behind human augmentation.

Using robotics for human augmentation can lead to a number of improvements. A manipulator can increase the size of the operator's workspace through its longer reach or can be used in a payload assist situation. In one implementation of payload assist, the manipulator grasps and carries a heavy load while the operator guides the manipulator by hand by pushing or pulling on its EEF. Payload assist is particularly associated with

designs for leg exoskeletons that allow humans to carry large payloads with greater ease and efficiency. [Walsh, 2006][Cao, 2009] Additionally, robots can perform payload assist and numerous other functions that aid the elderly and individuals with disabilities. [Harmo, 2005]

1.3.2 Flexible Manufacturing

Automated robotic systems for manufacturing have been used for decades. They perform tasks such as welding, painting, cutting, grinding, assembly, and machine loading. These jobs are often repetitive and the use of robots leads to lower labor costs and higher reliability. Commonly, the system automation is preprogrammed for well-defined tasks that must be repeated indefinitely with some degree of precision. Additionally, the same components are always handled and processed in the same order in a static environment.

A flexible manufacturing system would support a wide variety of tasks. Should a process change, less work would be required to get the system up and running a new process. Decreased downtime means less money lost for the manufacturer. Flexible manufacturing would allow the handling of various sizes of objects by the same manipulator. Thus, the order of steps in a process could vary, as could the sub-tasks of each step. A human worker could also work side-by-side with a robot with greater ease because it is programmed to handle uncertainties in the environment.

Ford has successfully demonstrated a system capable of producing three distinct models on one vehicle platform. [Venables, 2006] Due to the integration of vision sensors that detect roof dimensions, robots can identify the vehicle being produced and properly center the roof for welding. Flexible automation is particularly useful in application areas with a wide variety of low-volume tasks such as manufacturing

products for specific, short-term government or commercial projects. With many sectors such as automotive and aerospace using electronics, flexible manufacturing would also aid the low-volume demand for chips with similar characteristics that occur in the microchip industry. [Venables, 2005]

1.3.3 Nuclear and other Applications in Hazardous Domains

Radioactive environments present a specific type of hazard to workers. In these environments, minimizing the dose received by workers is of utmost importance. Shielding, increased separation distance from a radioactive source, and decreased time in the environment are all common techniques for reducing worker dose. Although robotics is also a means to alleviate this potential danger, in many applications it seems that the previously mentioned techniques provide sufficient protection. However, some materials and processes are too radioactive/hot for radiation workers and automation must be implemented. Finding the most practical solutions in such situations necessarily involves the fusion of principles from both mechanical engineering and nuclear engineering/science. In other words, there are design trade-offs made between automation and nuclear solutions when designing such a system. Several system requirements stem from the following areas:

- *Mechanical engineering*
    - Mechanical design: manipulators, layout, etc
    - Operational control: software, degree of system autonomy, etc
    - Maintenance

- *Nuclear engineering/science*
    - Shielding and material properties
    - Process modeling
    - Containment, safety, and security
    - Waste disposal

Installing robotics in radioactive environments is very challenging because of these cross-disciplinary requirements. For example, characterizing the radiation field can be difficult, and the exact effects that the environment has on the mechanical and electrical components of systems is hard to determine. A more difficult certification process for mechanical components also exists. Furthermore, unique requirements could eliminate systems that would be suitable and/or preferred if the processing occurred in a non-radioactive environment.

Often systems are constructed with very specific applications and requirements in mind, so an all-encompassing approach when employing robotics in these types of environments does not exist. However, there are still some general guidelines that can be established from past experiences to help govern the development of robotic systems for future applications. The challenge still remains to develop a broader and more generalized automation approach for these applications.

1.3.3.1 Glovebox Automation

When necessary, radiation workers commonly handle radioactive materials through some interface such as a glovebox.

Figure 1-1: Glovebox for radioactive material handling at Los Alamos National Lab [nuclearweaponarchive.org]

Automation in this environment is difficult because of environmental constraints, the geometry of the glovebox, and the additional requirements for handling nuclear materials. Despite these difficulties, glovebox automation has been demonstrated in the Advanced Recovery and Integrated Extraction System (ARIES) at Los Alamos National Lab (LANL) which is described in Section 3.1.1. Below are some of the general goals that glovebox automation must meet:

Minimize worker radiation exposure

Although radiation dose limits are set for radiation workers, the idea has always been that any dose should be avoided if possible. This is the idea behind the ALARA principle: that dose should be kept "As Low As Reasonably Achievable." Such is the motto adopted for interactions between workers and radioactive environments. The effects of the accumulation of low levels of

radiation are not well-defined, so a conservative approach is always taken when dealing with humans and radiation.

Minimize hands-on processing

There are several drawbacks to glovebox work that drive the effort to limit hands-on processing when possible. Extremity dose is a major concern as hands and arms are commonly the closest to unshielded radioactive materials. Shielded gloves are used to reduce this dose, but the gloves reduce operator dexterity. This makes some tasks more difficult and time-consuming to accomplish than if outside the environment, further increasing exposure time and dose. Complications are increased if two workers have to work together to complete a task. Ergonomic issues also exist and occasionally result in injuries that force workers to cease their glovebox activities until injury-free. From a processing perspective, worker capability issues have the combined effect of limiting process efficiency, capacity, and productivity.

One reason that nuclear materials are processed in gloveboxes is for containment purposes, and one of the easiest ways to breach containment is glove punctures. Due to this, pinch points and sharp edges must be avoided and sharp objects are often prohibited in the workspace. This may mean that the best tool or instrument for a task may be removed and the process restructured using less desirable components.

Non-proliferation and control of radioactive materials is also of utmost importance. Whenever there is direct access to nuclear materials, there is a greater chance that the materials could potentially be removed by an individual

who seeks to use them for harm. Thus, if only robots handle the materials directly, human access, and thus the threat of proliferation, is decreased.

High system stability and process repeatability

The nature of the materials and objects being handled demands a high level of safety that can be directly correlated to system stability (i.e. the system can conduct all processes without unexpected responses). If materials are dropped, it requires extensive clean up, and any breach in containment generated by the robot is unacceptable. The goal of limiting the human presence in the environment cannot be met if a system continually breaks down, requiring additional maintenance and manual processing to complete unfinished tasks.

Sometimes manufacturing processes are done where quality requirements must be met regardless of the environment, including precise measurements of geometry and mass. If quantities of a product are fabricated by different workers, this may lead to product variation which would be limited by the process repeatability of a robot. Due to the manufacturing parallels, the benefits of flexible manufacturing can also be realized in gloveboxes.

1.3.3.2 Hotcell Automation

Hotcells are used for handling radioactive materials that are too hot for glovebox environments. Due to the hazard, the worker is separated from the nuclear material by a few feet of shielding materials, including concrete and leaded glass. Automation and/or robotics have been used more extensively in these applications, and the robotic systems deployed are often cable-driven master-slave systems.

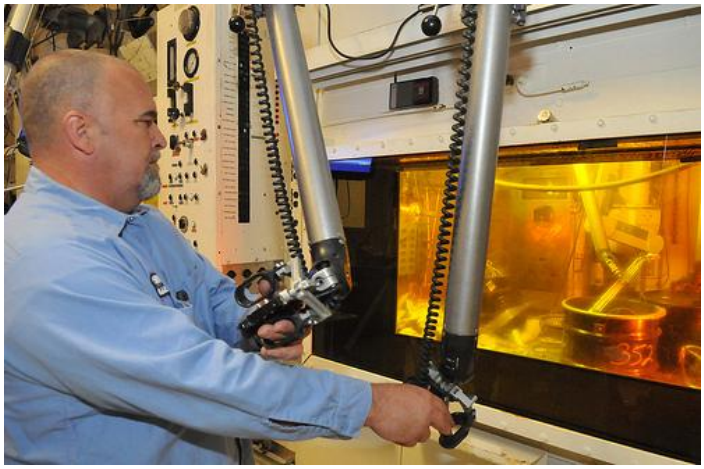


Figure 1-2: Packaging Remote-Handled TRansUranic waste (RH-TRU) using a hotcell. [Argonne National Laboratory, 2009]

The operator is very involved in the control loop, and the successful completion of a task often depends on the competency of the worker in physically controlling the robot. Thus, hotcells and gloveboxes share the common dependence on the abilities of the workers and the difficulties that stem from this. Additionally, technologies that are first developed in gloveboxes have the capability to extend to hotcell applications.

1.3.3.3 Decontamination and Decommissioning

Decontamination and Decommissioning (D&D) of nuclear facilities can extend over a long period of time. Thus, considerable amounts of radiation dose could be received by workers. The nature of the work will dictate the type of robotic system needed, with mobile systems often deployed for processes that are more remote. Various types of systems have been deployed for D&D of storage tanks, containment structures, and reactor and process facilities. [Harden, 2009][Seward, 2005][Noakes, 2000]

D&D work can present challenges in regard to task specification and environment. In locations where contamination has occurred, the type of scenario cannot always be predicted, so the robotic system needs the ability to handle whatever conditions exist. Even when the task is somewhat well-defined, some issues may not be apparent until the system reaches the target area. Additionally, rooms and various facilities are structured differently. Due to all this potential variation, it would be more cost efficient to have a flexible system capable of multiple D&D tasks rather than creating a new system for each task.

1.4 SUMMARY

We have discussed some of the considerations when developing robotic systems. Many system requirements are derived from the tasks to perform and the environmental constraints. These requirements also play an important role in the level of system autonomy and the type of robotic system employed. For some of our application areas, especially in the nuclear domain, we have unique constraints that make automation desirable, but difficult to implement. Through this research, we will address grasping configurations and dexterity in the workspace to develop incremental steps in system autonomy that aid the use of more advanced robotic systems in our application areas.

Chapter 2 **Kinematic Modeling**

Kinematics is the study of robotic motion, analyzing their position, velocity, and acceleration. In order to conduct this analysis, we need a means of modeling manipulator kinematics. To develop this model, this chapter addresses the methods used in robotics research to describe robot geometry and their spatial locations relative to other objects of interest. A background knowledge in linear algebra is necessary, and although not developed here, a more thorough description of the mathematics is found in Pryor and Tesar [1999].

Particularly, a kinematic model allows us to develop a relationship between the joint angles and the EEF's position and orientation, which is essential to robot control. We will focus on the kinematic model and not other manipulator models such as those for dynamics and compliance because our work is most concerned with manipulator configurations. This formulation is also necessary for utilizing redundant systems and for executing collision detection and singularity avoidance, topics of later discussion.

2.1 FORWARD KINEMATICS

The kinematics of a robot consists of the relative motion of its links. This motion is produced by displacements in the joints of the robot. For an n degree-of-freedom (DOF) robot, the joint vector ϕ is given by

$$\phi = \phi(t) = (\phi_1(t), \phi_2(t), \ldots \phi_n(t))^T. \tag{2-1}$$

The EEF's location $\boldsymbol{u}$ is uniquely determined by ϕ. Finding the vector $\boldsymbol{u}$ is known as the forward kinematics problem which determines the position and orientation of the EEF based on the joint angle displacements.

2.1.1 DH Parameters

In order to perform forward kinematics, we need to model the displacement of joints relative to one another. This can be done using the Denavit-Hartenberg (DH) parameters which provide a representation of manipulator geometry. [Craig, 1989] The four DH parameters (a, α, d, θ) represent the minimal geometric representation for a serial chain manipulator of one DOF joints. They are used to produce the coordinate transformations between joint axis frames. Particularly, using the DH parameters, we can perform multiple linear transformations to locate the EEF in the global frame. Instead of using six parameters to uniquely define a frame, Denavit and Hartenberg were able to use four parameters by imposing two constraints on the frames:

- The z-axis of each frame must be along the axis of rotation for a revolute joint or in the same direction as the translational motion for a prismatic joint, and
- The x-axis of frame *i-1* must intersect and be perpendicular to the z-axis of frame *i*.

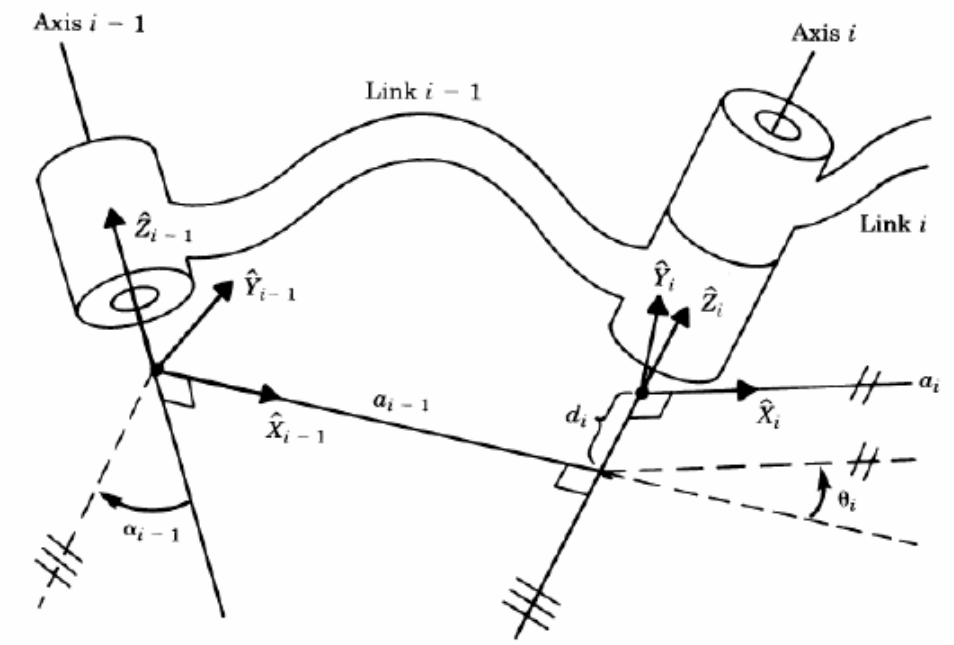


Figure 2-1: DH parameter formulation [Craig, 1989]

Based on Figure 2-1, the DH parameter notation is as follows:

- a_i is the distance from $\hat{Z}_i$ to $\hat{Z}_{i+1}$ measured along $\hat{X}_i$
- α_i is angle between $\hat{Z}_i$ and $\hat{Z}_{i+1}$ measured about $\hat{X}_i$
- d_i is the distance from $\hat{X}_{i-1}$ to $\hat{X}_i$ measured along $\hat{Z}_i$
- θ_i is the angle between $\hat{X}_{i-1}$ and $\hat{X}_i$ measured about $\hat{Z}_i$

Some of these parameters are variable while others are fixed. The parameter θ_i is variable for rotary Joint *i*, and d_i is variable for a prismatic Joint *i*. The number of variable parameters is equal to the DOF of a serial manipulator. With this formulation, a four by four homogeneous transformation matrix can be used to move from a frame on Joint *i* to a frame on Joint *i-1*. A homogeneous matrix allows for the translations between

frames to be included in the transformation. The transformation from frame *i* to frame *i-1* is given by the following matrix,

$$ {}^{i-1}_{i}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i \cos\alpha_{i-1} & \cos\theta_i \cos\alpha_{i-1} & -\sin\alpha_{i-1} & -d_i \sin\alpha_{i-1} \\ \sin\theta_i \sin\alpha_{i-1} & \cos\theta_i \sin\alpha_{i-1} & \cos\alpha_{i-1} & d_i \cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2-2} $$

Thus, each transformation between frames is built from the four DH parameters, including those which are variable. For an *n*-DOF serial system, consecutive transformations between frames can be made to transform frame *n* on the EEF to the base frame,

$$ {}^{0}_{n}T = {}^{0}_{1}T\,{}^{1}_{2}T \ldots {}^{n-1}_{n}T. \tag{2-3} $$

If the base frame is taken as the global frame, after transformation, we have the location of the EEF in terms of the global (Cartesian) frame. There are two common DH parameters formulations, and either of these can be used to describe the kinematic models in our work. Note that joint limits are not built into this formulation and must be implemented elsewhere.

2.1.2 First Order Influence Coefficients (the Jacobian matrix)

As described, forward kinematics can be used to uniquely determine the robot EEF. Generally, a fixed point of reference from the EEF, or the tool point, is the point of interest. This same formulation, however, can be used to describe the position of any

point on the manipulator. In Cartesian space, six parameters are usually needed to fully specify the position of a manipulator's EEF/tool point,

$$\boldsymbol{u} = (x, y, z, \psi_x, \psi_y, \psi_z)^T. \tag{2-4}$$

The first three parameters give the tool point location in space, and the second three parameters determine the tool point orientation. The first derivative of the Cartesian position vector with respect to time defines the velocity vector

$$\dot{\boldsymbol{u}} = (\dot{x}, \dot{y}, \dot{z}, \dot{\psi}_x, \dot{\psi}_y, \dot{\psi}_z)^T. \tag{2-5}$$

Thus, the first three parameters describe the translational velocity of the tool point and the other three parameters describe the angular velocity. Similarly, the acceleration vector can be obtained by differentiating the position vector twice with respect to time.

As mentioned, the location and orientation of the EEF in Cartesian coordinates is a function of the joint angles,

$$\begin{aligned} x &= f(\phi_1, \phi_2, \ldots \phi_n), \\ y &= f(\phi_1, \phi_2, \ldots \phi_n), \\ z &= f(\phi_1, \phi_2, \ldots \phi_n), \\ \psi_x &= f(\phi_1, \phi_2, \ldots \phi_n), \\ \psi_y &= f(\phi_1, \phi_2, \ldots \phi_n), \\ \psi_z &= f(\phi_1, \phi_2, \ldots \phi_n). \end{aligned} \tag{2-6}$$

Thus, taking the time derivative of the Cartesian vector $\boldsymbol{u}$ requires the chain rule,

$$\dot{\boldsymbol{u}} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\phi}} \frac{\partial \boldsymbol{\phi}}{\partial t} = \left[G_{\phi}^{u} \right] \dot{\boldsymbol{\phi}} = \boldsymbol{J} \dot{\boldsymbol{\phi}}$$

or

$$\dot{\boldsymbol{u}} = \boldsymbol{J} \dot{\boldsymbol{\phi}} . \tag{2-7}$$

The partial derivatives $\partial \boldsymbol{u} / \partial \boldsymbol{\phi}$ produce what are known as the first-order influence coefficients. The Jacobian $\boldsymbol{J}$ is built from these coefficients, and is used to map the joint velocities to end-effector velocities. The Jacobian is only a function of manipulator geometry, and thus only changes as the configuration of the manipulator changes. The Jacobian (i.e. the first-order influence coefficients) is also used to relate joint and EEF accelerations,

$$\ddot{\boldsymbol{u}} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\phi}} \frac{\partial^2 \boldsymbol{\phi}}{\partial t^2} + \frac{\partial^2 \boldsymbol{u}}{\partial \boldsymbol{\phi}^2} \left(\frac{\partial \boldsymbol{\phi}}{\partial t} \right)^2 = \left[G_{\phi}^{u} \right] \ddot{\boldsymbol{\phi}} + \dot{\boldsymbol{\phi}}^{\mathrm{T}} \left[H_{\phi\phi}^{u} \right] \dot{\boldsymbol{\phi}}, \tag{2-8}$$

where $\left[H_{\phi\phi}^{u} \right]$ is defined as the Hessian and determines Coriolis and centripetal effects. $\left[H_{\phi\phi}^{u} \right]$ also refers to the second-order influence coefficients, and, similar to the Jacobian, is solely a function of geometry.

The importance of these derivations lies in the description of the Jacobian. Not only does it help map joint velocities and acceleration to end-effector velocities and accelerations, but it gives an indication of the quality of joint configurations and whether a singularity has been encountered. From a control standpoint, singularities should be avoided because we lose the ability to effect motion in certain directions and the manipulator is unstable.

2.1.3 End-Effector Orientation

There are also a few different ways of describing the orientation of the end-effector. This can be given as three angle rotations, $(\alpha, \beta, \gamma)^T$, about the global frame that produce the tool frame orientation. In the Fixed XYZ representation, the rotations are about the fixed global frame axes. Thus, there is a rotation α about the fixed x-axis, a rotation β about the fixed y-axis, and a rotation γ about the fixed z-axis. Although this is a simple way to describe the orientation, it is sometimes difficult to visualize how a rotation about a fixed axis will change the end-effector orientation. [March and Tesar, 2004]

Euler angles can also be used to represent EEF orientations. These angles are made relative to a rotated frame. For example, an Euler XYZ representation gives a rotation α about the fixed x-axis, a rotation β about the rotated y-axis produced from the first rotation, and then a rotation γ about the rotated z-axis produced from the first two rotations. Euler angles can be defined as rotation about any order of the x, y, and z axes as long as no consecutive axes are the same. Thus, an Euler ZYZ is a valid representation for EEF orientation. More on Euler angles can be found in Craig [1989].

These representations are worth noting since we are concerned with dexterity in the workspace. Thus, we want a variety of EEF orientations to be possible at a point of interest. When testing for the level of dexterity, we will use these representations to sample various EEF orientations at a point in the workspace.

2.2 INVERSE KINEMATICS

For many of our applications, the ability to command the manipulator's EEF in free-space is crucial for completing a task. While control of the EEF is desired, the operator has direct control over the manipulator's joint angles ϕ. Given the position,

velocity, or acceleration of the EEF, inverse kinematics solves for the ϕ necessary for attaining that EEF pose (location and orientation).

2.2.1 Complications

Performing inverse kinematics is a more complicated process than forward kinematics. Since the Jacobian matrix maps from joint space to Cartesian space, an inverse operation can be performed to map from Cartesian space to joint space. A simplified representation of this is,

$$\boldsymbol{J}^{-1}\dot{\boldsymbol{u}} = \boldsymbol{J}^{-1}\boldsymbol{J}\dot{\phi},$$

or (2-9)

$$\dot{\phi} = \boldsymbol{J}^{-1}\dot{\boldsymbol{u}}.$$

Errors arise when the Jacobian is inverted at (or even near) a singular configuration. At a singularity, the robot effectively loses a DOF. One example is if two or more joints simultaneously produce the same motion. Since the quality of a configuration is related to the state of the Jacobian, the Jacobian is used to define many metrics. For example, a Jacobian determinant of zero occurs when a manipulator is in a singular configuration. Thus, tracking the determinant can be an indication of the state of the system. However, even though the determinant may be close to zero, the manipulator may not be moving toward a singular configuration. A better metric is the *condition number* κ of a matrix $\boldsymbol{A}$,

$$\kappa = \|\boldsymbol{A}\| \cdot \left\|\boldsymbol{A}^{-1}\right\|, \tag{2-10}$$

where $||\boldsymbol{A}||$ is the norm of matrix $\boldsymbol{A}$. The condition number describes whether a linear system is *well-conditioned* or *ill-conditioned*. A system is ill-conditioned when κ is very large, meaning that a small change in the input vector leads to large changes in the output vector. [Pryor and Tesar, 1999] In our case, we are interested in the conditioning of the Jacobian, where its condition number is

$$\kappa_J = \|\boldsymbol{J}\| \cdot \|\boldsymbol{J}^{-1}\| . \tag{2-11}$$

An ill-conditioned Jacobian produces large EEF velocity changes given small changes in joint velocities or small inaccuracies have a large effect on the EEF. This is a problem from a control perspective and one of the reasons why singularities are avoided.

There are also some EEF poses that can be attained by more than one joint configuration, so a unique joint angle solution is not guaranteed when performing inverse kinematics. Different branches of solutions arise, and control systems need to limit solutions to a single branch so that the EEF does not "jump" between branches. Redundant systems, which will be discussed in the next section, further complicate the inverse as an infinite number of solutions exist.

2.2.2 Redundant Systems

In the case of redundant systems, there are additional complexities to inverse kinematic calculations. A redundant manipulator possesses a higher DOF than that needed to uniquely map to an EEF location in free space or perform a task. Thus, a manipulator in three-dimensional free space is redundant if it has more than six DOFs. For inverse calculations, the pseudoinverse is often used. For more on the use of the pseudoinverse for redundant systems, refer to [Pryor and Tesar, 1999] and [Klein, 1985].

Performing inverse kinematics on a redundant system generates an infinite number of joint solutions. Once joint options are generated, a method is needed to determine which solution to select. The decision-making process is usually done with criteria that measure the performance or behavior of some aspect of the manipulator. A decision-making system is needed to select joint configurations based on the chosen criteria and their respective weights in the decision. Pryor and Tesar [1999] discuss common decision-making schemes or *Redundancy Resolution Techniques* (RRTs).

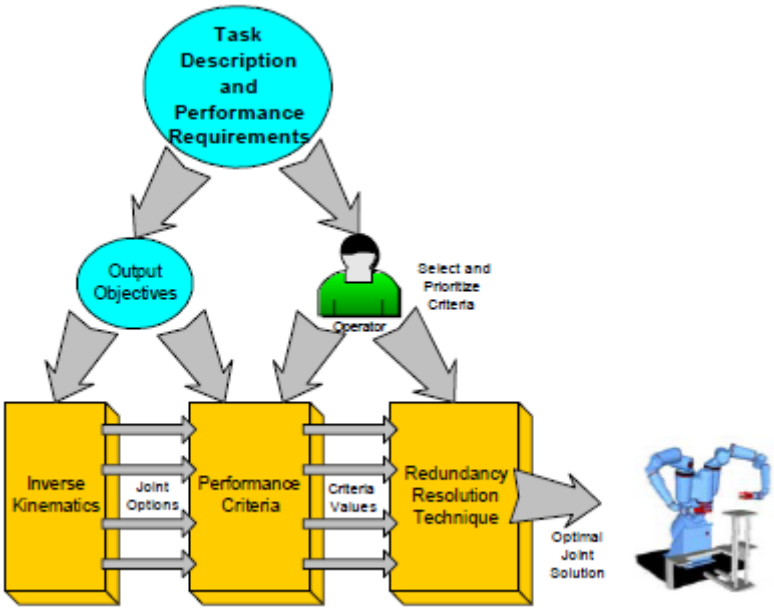


Figure 2-2: Scheme for utilizing manipulator redundancies [March and Tesar, 2004]

Criteria can be selected based on the optimization of some quantity or on the task. In regard to kinematics, some criteria involve the Jacobian, such as the condition number and the Measure of Transmissibility,

$$\gamma_{MOT} = \sqrt{\det(JJ^T)}. \tag{2-12}$$

The Joint Range Availability (JRA) criterion chooses configurations that best keep the manipulator away from joint limits,

$$\gamma_{JRA} = \min_i \left(1 - \frac{\left| \theta_i - \theta_{i,mid} \right|}{\theta_{i,\max}} \right), \tag{2-13}$$

where θ_i is the joint displacement, $\theta_{i,mid}$ is the displacement at the midpoint of the travel range, and $\theta_{i,max}$ is the displacement at the travel limits of Joint i.

Redundancies can also give the system the ability to perform secondary objectives. In regard to motion planning, obstacle avoidance can be performed, where redundancies are utilized to select joint configurations that best move the manipulator away from obstacles/potential collisions. Criteria-based decision-making will be addressed more in Chapter 3.

We will use some redundant systems in our work. To make use of dexterous regions, we also employ redundancies in the form of task requirements that reduce the number of constraints, leading to an additional DOF for the task. For example, sometimes the grasping direction on an object is not important which means one of the angles used to describe the EEF orientation may be free. Then, the JRA criterion could be used to determine which direction/free angle is optimal in regard to joint limits.

2.3 SUMMARY OF KINEMATIC MODELING

In this chapter, a kinematic model was described for serial manipulators. This formulation allows us to perform forward and inverse kinematics for a system. We also discussed some indicators of poor configurations and the use of criteria to help select preferable configurations. One of our goals is to create a more reliable system that can

make decisions to avoid unfavorable configurations. In the system control structure, it is also important to include dynamic effects on a system, the behavior of a system under load, system compliance, etc. However, we are just concerned about what kinematic configurations should be attained, with other system models being used in the control process. This restriction to kinematics will also guide our discussion of previous work.

Chapter 3 **Previous Work**

Developing the area of grasping requires knowledge collected from a variety of robotic disciplines, including task planning, layout analysis, modeling, and motion planning. Work in these areas is extensive, and although it is not an exhaustive list, we will survey work conducted by the Robotics Research Group (RRG) and other researchers that is related to these fields and relevant to the research objectives stated in Section 1.2.

3.1 TASK PLANNING & LAYOUT ANALYSIS

One of the first steps to developing an automated system is to define the tasks and sub-tasks that must be executed to carry out a process. This may first involve identifying the high level objective, for example, "Weigh the selected canister." This task is then broken down into a series of basic commands that the robot can understand, such as, "Move robot end-effector from Frame A to Frame B," "Close gripper to grasp canister lid," etc. Eventually, an automated system may be able to take those higher level commands and define the necessary sub-tasks or inform the operator that the task is not possible given the current state of the manipulator and the environment. Such a system is beyond the scope of this effort, but our research objectives will partially reduce the burden on the operator and progress our operational capabilities toward this goal.

Various process requirements stem directly from the task analysis. These can include quantifying the necessary robot payload, EEF tooling, velocity and acceleration limits, etc. Other factors that need to be considered include constraints imposed by robot geometry, physical constraints of the workspace like its dimensions, and environmental requirements. These requirements will help determine a robotic system's attributes.

The task analysis also drives the layout analysis of the system, also referred to sometimes as configuration management. [LeGoullon and Tesar, 1997] The choice of manipulator plays a major role in the configuration of task targets in the workspace and the placement of the manipulator itself. The layout analysis is enhanced by a clear understanding of a manipulator's motion capabilities. Thus, much work has been conducted to identify, use, and improve these capabilities. Once a layout is selected, simulations are commonly performed to demonstrate how the system functions for a particular layout.

3.1.1 ARIES Analysis

The automation of the Advanced Recovery and Integrated Extraction System (ARIES) at LANL is a specific example of the task analysis process. This plutonium processing line takes the pits from old nuclear weapons and converts them to a form suitable for re-use or long-term storage. The RRG provided an initial task and layout analysis of a system for automated plutonium processing. [Cox, 1999]

The ARIES plutonium processing takes a finite number of steps from beginning to end. The entire process was sub-divided into eight modules, and five gloveboxes were used according to their function in the line: Pit Bisection, Hydride/Dehydride, Hydox, Canning and Welding, and Decontamination. The low-level robotic tasks performed in these gloveboxes were then placed into one of five task categories: Material movement, Material Positioning and Orientation, Process Control and Coordination, Sensing, and Inspection. The number of task types per glovebox was also compiled. [Cox, 1999]

Looking specifically at the Canning and Welding glovebox, a number of sub-tasks were defined (see Figure 3-1).

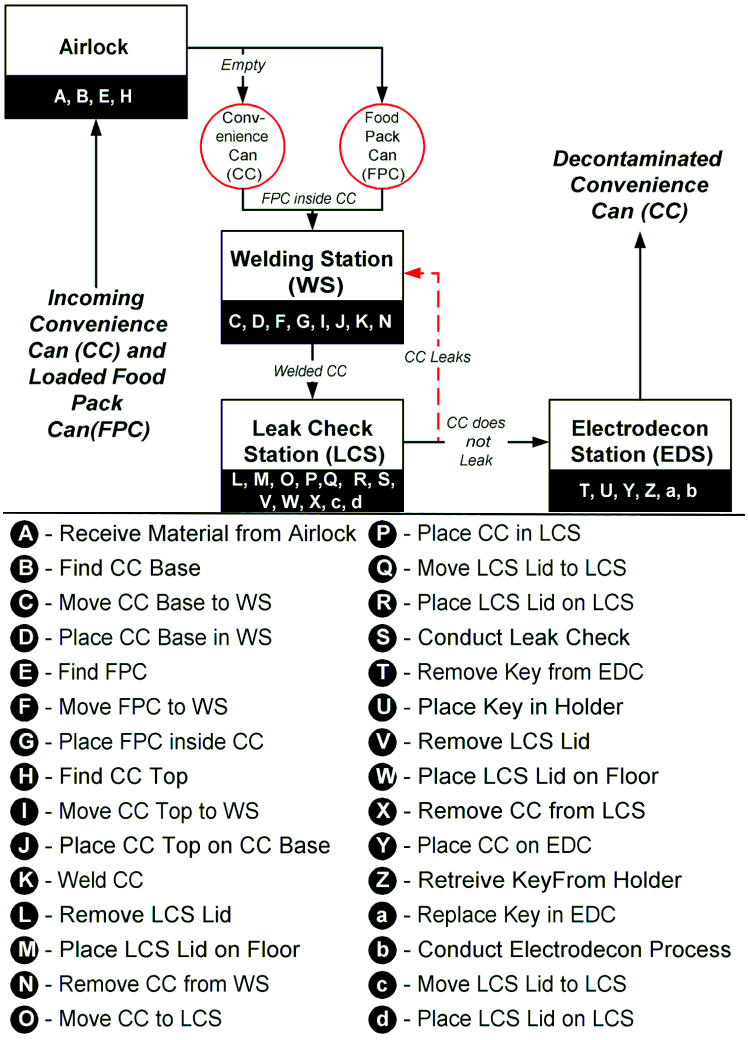


Figure 3-1: The task analysis of the Canning and Welding ARIES glovebox. [Kapoor, 2002]

Tasks were grouped and assigned to one of a few stations within the glovebox workspace. These stations and the manipulator were then positioned so that all of the tasks could be completed. The resulting layout is seen in Figure 3-2. One process of selecting a feasible layout is discussed in the next section.

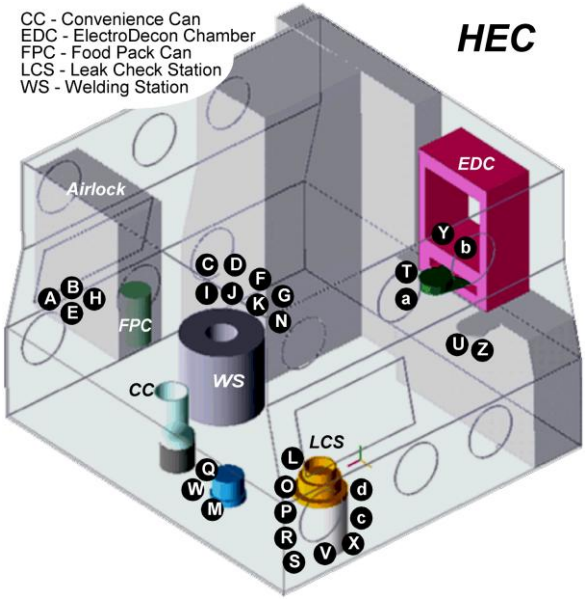


Figure 3-2: The layout of the Canning and Welding ARIES glovebox. [Kapoor, 2002]

3.1.2 Configuration Management

Configuration management [LeGoullon and Tesar, 1997] can be thought of as a complex design optimization problem. The goal is find the best way to use all the resources at hand to complete a task. This involves selecting the proper components to use, how to orient each component individually, and also how to arrange components into an optimal configuration for the given process/task. The situation becomes increasingly complicated as more tasks are supported. In this case, a configuration that optimizes one task may not support another, so another configuration must be found that optimizes the combination of both tasks.

Traditionally, configurations were selected based on intuition and trial-and-error. Eventually, a solution could be found that was capable of completing the task, but it was unlikely that this configuration was the most efficient or enabled the highest system

performance. Having a method to select system layouts would reduce design time and provide an optimal solution.

LeGoullon and Tesar [1997] used two approaches to configuration management: criteria-based decision making (in Section 2.2.2) and manipulator design guidelines [Hill and Tesar, 1997]. The use of criteria allows the designer to quantitatively compare the quality of different configurations. The manipulator design guidelines are used to determine the mechanical properties of the robot that are necessary to achieve the desired level of robot performance.

LeGoullon and Tesar [1997] applied these tools to address robot workcell configurations, particularly workcells for airframe assembly. Five different workcell layouts were proposed and each was progressively more advanced and flexible to increase the percentage of airframes that could be assembled. These layouts were then analyzed using multiple criteria and manipulator design guidelines. The selected criteria values were calculated, normalized, and weighted to create an overall performance index that was utilized to select the optimal workcell configuration. This evaluation was restricted to five unique configurations in order to deal with complexity of the air frame assembly application. Conversely, our effort will consider a smaller and simpler set of tasks that allow for the possible consideration of a continuous spectrum of configuration options.

Additionally, LeGoullon and Tesar examined a variety of criteria in multiple categories, shown in Table 3-1. For this research, the focus will be on parametric and configuration criteria, since the task space does not require high torques, velocity, or other advanced performance evaluation – the robotic system operational parameters are likely to exceed the task requirements by large margins.

Table 3-1: Categories and descriptions for performance criteria [LeGoullon and Tesar, 1997]

Criteria Category	**Description**
Parametric	Based on the robot's physical limitations
Dynamic	Based on the dynamic model of the robot
Configuration	Based on the robot's Jacobian matrix
Obstacle Avoidance	Based on models of workspace obstacles
Compliance	Based on the compliances of the robot's links and joints
Fault Tolerance	Based on the robot's ability to survive a joint failure

Workcells are designed to be reconfigurable in order to support various batches of operations. In the long-term, however, without configuration management, new workcell layouts could be found, but higher levels of performance and flexibility would not be optimized.

3.1.3 MOX Fuel Fabrication

The RRG has also completed work on an initial conceptual design for an automated system for mixed-oxide (MOX) fuel fabrication at Los Alamos National Laboratory (LANL). [Williams, 2009] At LANL, MOX-type fuels, which consist primarily of uranium and plutonium oxides, are being developed for testing in the Advanced Test Reactor (ATX) in Idaho. These test fuels also contain small amounts of minor actinides such as americium and neptunium. These are added to determine their effects on fabrication parameters and in-reactor performance as they are transmuted in the reactor. Currently, these fuels are fabricated by hand inside of glovebox environments,

and many benefits could be realized through the incorporation of robotics into the fabrication process as discussed in Section 1.3.

The task analysis examined all the steps of the MOX fabrication process as they are currently performed at LANL. These steps include powder mixing/milling, pellet pressing and sintering, and post sintering treatment/analysis. There are also intermediate weighing and sampling steps for quality assurance and material accountability. We narrowed our focus to the mixing/milling stage which requires powder measuring and milling. For this process, we considered the components necessary to complete these tasks: manipulators, fixtures, containers, instruments, etc. Then the system level tasks associated with the mixing/milling stage were grouped into categories: Measuring, Milling, Sampling/Pellet Measuring, Tool Change, and Miscellaneous.

Table 3-2: Task analysis of MOX glovebox [Williams, 2009]

Abbreviations	
CR = crucible MC = milling container PC = powder can SC = shielding container SCT = scoop tool	ML = mill location MP = mixing position OL = opening location STT = storage table TSL = tool storage location

Measuring	**Milling**	**Sampling/Pellet Measuring**
A. Lift SC lid B. Move lid to storage shelf C. Move PC from SC to OL D. Remove lid from PC E. Move lid to STT F. Move open PC to scale area G. Pour powder from can into CR for coarse measurement H. Move PC to OL I. Use SCT to extract powder from can for fine measurement J. Move scoop from PC to CR K. Measure/empty powder into CR using SCT L. Obtain lid from STT M. Place lid on PC N. Move PC to SC O. Move SC lid from storage shelf P. Place lid on SC	Q. Move MC from storage to MP R. Remove lid from MC S. Place lid on STT T. Move CR from scale to MP U. Pour powder from CR into MC V. Grab milling ball from STT W. Add milling ball to MC X. Obtain lid from STT Y. Place lid on MC Z. Move MC from MP to ML * Operator receives MC and runs the mill a. Move MC from ML to MP R. Remove lid from MC S. Place lid on STT b. Move intermediate can from storage to OL c. Move MC to pouring location d. Empty milled contents into intermediate can Q. Move MC to MP	e. Move sample CR from STT to scale I. Extract milled sample using SCT J. Move SCT from can to CR K. Measure/empty sample into CR f. Move CR from scale to STT **Tool Change** g. Move to old TSL h. Unload old tool i. Move to new TSL j. Attach new tool k. Move from TSL **Miscellaneous** l. Open side hatch m. Move can from side port to STT n. Close side hatch o. Move container to scale for weighing p. Move container from scale

Once the sub-tasks were defined, a layout analysis was conducted. The layout analysis did not focus on finding an optimal configuration but rather on finding a valid layout that could be used in simulation to conceptually demonstrate how a system could carry out the desired tasks. The results from the task and layout analyses are shown in Figure 3-3. Future work could involve using previous methods and those developed in this research to determine an optimal configuration within the glovebox for this process.

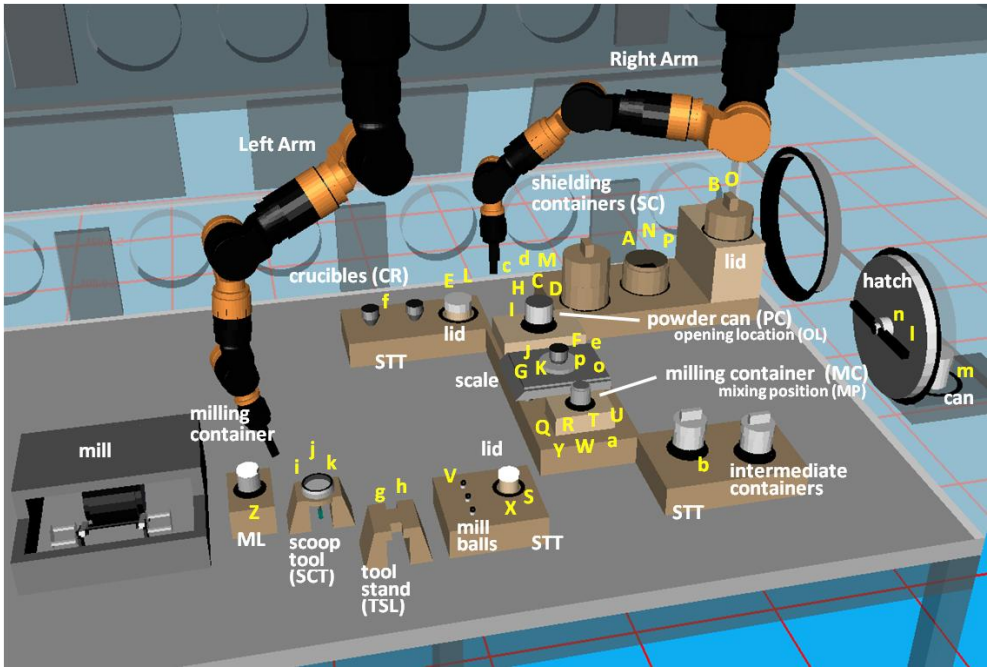


Figure 3-3: Layout analysis of MOX glovebox showing position of tasks and work locations listed in Table 3-2 [Williams, 2009]

3.1.4 Maximizing Dexterity

As mentioned, the layout analysis/configuration management of a system can be construed as a design optimization problem. Work has focused on maximizing manipulator dexterity in the workspace or utilizing the more dexterous regions of the

workspace to execute tasks.[3] Maximizing dexterity allows the manipulator to perform a larger number of tasks and also perform these tasks away from singularities. Much of the work presented here deals with analyzing a system after the physical dimensions of the robot have already been selected, but the criteria can also be used during the design and layout analysis.

3.1.4.1 Configurations/Postures

Chiu [1988] proposed using velocity and force ellipsoids to select manipulator postures that are compatible with task requirements. These Jacobian based velocity and force ellipsoids represent the velocity and force transmission characteristics given by a particular manipulator posture. The velocity ellipsoid is defined by

$$\dot{x}^T (JJ^T)^{-1} \dot{x} \leq 1. \tag{3-1}$$

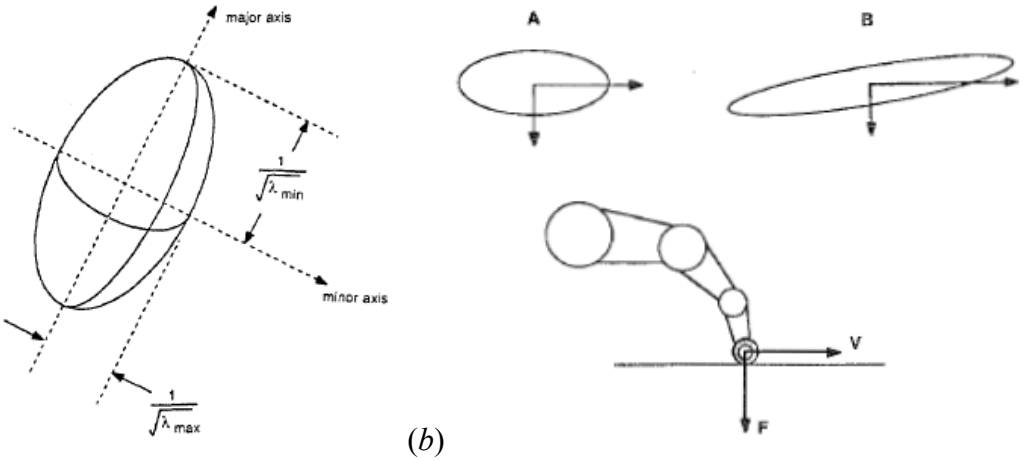
where $\dot{x}$ are the task coordinate vectors and J is the Jacobian that maps the joint velocities to the task velocities. Similarly, the force ellipsoid is defined by

$$f^T (JJ^T) f \leq 1, \tag{3-2}$$

where f is the force vector in task space and the same Jacobian maps forces in joint space and task space. The principle axes of the velocity and force ellipsoids correspond to the reciprocal of the square root of the eigenvalues of $(JJ^T)^{-1}$ and (JJ^T), respectively. The eigenvalues are reciprocals of one another, so the principal axes of the ellipsoids

[3] We will group the use of the dexterous workspace and the avoidance of joint limits and singularities into techniques for increasing manipulability in the workspace. At times, manipulability and dexterity may be used interchangeably.

coincide, but the lengths of the axes are in inverse proportions. [Yoshikawa, 1985] Thus, the major axis of the velocity ellipsoid is the optimal direction for effecting velocity and the minor axis is the optimal direction for accurately controlling velocity, and vice versa for the force ellipsoid for transmitting or controlling forces. The shape of the velocity and force ellipsoids is changed by alternating the manipulator's posture (see Figure 3-4). This effectively changes the optimal direction for effecting or controlling velocity or force.



(*a*) (*b*)

Figure 3-4: The ellipsoid approach developed by Chiu [1987]. Image (*a*) gives the ellipsoid formulation. Image (*b*) compares the force ellipsoids for two different postures. The transmission ratio along a particular direction is determined by the intersection of the directional vector with the ellipsoid surface.

The transmission ratios represent the amplifications in the magnitude of velocity and force when mapping from joint space to EEF space. The force and velocity transmission ratios are given as

$$\alpha = [u^T (JJ^T) u]^{-1/2} \text{ and } \beta = [u^T (JJ^T)^{-1} u]^{-1/2}, \tag{3-3}$$

respectively, where u is a unit vector in the direction of interest. The velocity and force transmission ratios are greatest along their respective major axis, and are smallest along the minor axes. Using the transmission ratios, Chiu [1988] derived the *compatibility index* as

$$c = \sum_{i=1}^{l} w_i \alpha_i^{\pm 2} + \sum_{j=l+1}^{m} w_j \beta_j^{\pm 2}, \tag{3-4}$$

where w_i and w_j are weighting factors that indicate the relative magnitude or accuracy requirements in the respective task directions. Determining the optimum manipulator posture for a task is a matter of finding the posture that maximizes the compatibility index.

This index can also be used to find the optimal location in the workspace for a task with a particular velocity or force requirement. To do this, the compatibility index is calculated for different EEF positions, and the optimum index identifies where the task is best completed in the workspace relative to the position of the manipulator. The optimization of the compatibility index is also one way of utilizing manipulator redundancy. [Chiu, 1987]

Simulations of a planar manipulator led to postures which resembled anthropomorphic postures for completing a task with similar velocity and force requirements. It is possible then that optimization of such an index may be one factor in developing a scheme to mimic natural human arm postures. This may be a useful given that, in our research, mimicking the way humans grasp objects may provide a good baseline in the decision making process.

3.1.4.2 Placement of Manipulators

The work of Abdel-Malek [2004] examines the optimal placement of a manipulator base relative to targets in a predefined environment. Using the methods presented, manipulator placement can be used to configure a system that can reach a number of targets with maximum dexterity. The researchers develop a new dexterous performance metric,

$$W_d = \sqrt{\left| G_{z^*} G_{z^*}^T \right|}, \tag{3-5}$$

where G_{z^*} is the augmented Jacobian matrix given as

$$G_{z^*} = \begin{pmatrix} \Phi_q & 0 \\ I & q_\lambda \end{pmatrix}. \tag{3-6}$$

This is a $(n+3)\times(2n)$ matrix, where n is the DOF in the system. Also, $\Phi_q = \partial\Phi_i / \partial q_i$ is a $(3\times n)$ matrix, I is the identity matrix, and $q_\lambda = \partial q_i / \partial \lambda_j$ is an $(n\times n)$ diagonal matrix with

$$q_i = ((q_i^L + q_i^U)/2) + ((q_i^U - q_i^L)/2)\sin\lambda_i, \tag{3-7}$$

where q_i^L and q_i^U are the lower and upper limits of Joint i and λ_i is a slack variable for Joint i used for optimization purposes.

Thus, this type of Jacobian combines information about the position, orientation, and joint limits of the EEF. This measure lacks the drawbacks of previous measures. For instance, although the manipulability ellipsoid approach is widely used, it "does not

transform the exact joint velocity constraints into task space and so may fail to give exact dexterousness measure and optimal direction of motion in task space." [Abdel-Malek, 2004]

Finding the manipulator placement that maximizes dexterity is subject to a few constraints. The known dimensions and range of motion of the robot are used to generate the reachable envelope of the robot. In order to drive the workspace toward the target, two constraints are imposed: the target points are inside the workspace volume and the target points must be at least a specified distance from the workspace boundary (i.e. not on the boundary).

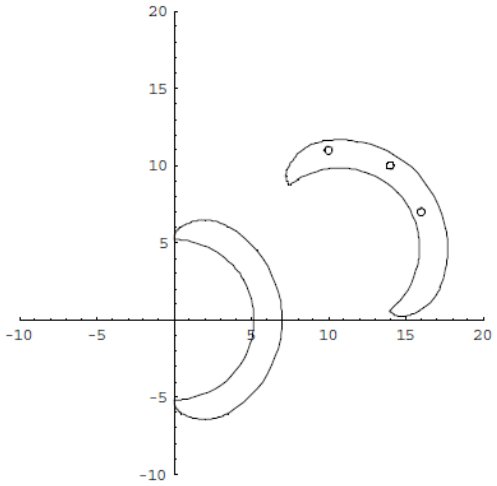


Figure 3-5: The initial and final locations of a workspace for an arm modeled by three revolute joints in the plane. [Abdel-Malek, 2004] The final location of the workspace is shown to contain the three targets of interest (shown as circles).

A cost function calculates the dexterity measure W_d at each target point relative to the boundary and then sums all measures multiplied by their specified weights. The cost function is calculated for a position of the workspace boundary. An iterative algorithm then moves the workspace and recalculates the cost function if the target points

are not greater than a specified tolerance inside the workspace. In the end, this method finds a locally optimal location for the manipulator base in the workspace that takes into account singular behavior and joint limits and does not involve inverse kinematics.

3.1.4.3 Redundancies

Redundant manipulators allow for the selection of an optimal configuration given a single EEF location and orientation, leading to the execution of secondary objectives which are often performance related. Redundancies are also frequently used to avoid obstacles in the workspace, which will be covered in a future section. In our case, these redundancies can be utilized to obtain grasping configurations that optimize dexterity in the workspace.

Dubey [1988] looked at utilizing redundancies to choose configurations that lowered the joint velocities or joint torques necessary to achieve the desired velocities and forces at the EEF. They defined local performance measures for a robot as its ability to follow a specified EEF trajectory and apply desired forces at the EEF. The velocity and force desired at the EEF are task dependent.

The manipulator velocity ratio (MVR) is defined at the ratio of the EEF velocity norm to the joint velocity norm. The manipulator mechanical advantage (MMA) is defined as the ratio of the EEF force vector norm to the norm of joint torque vector. Dubey [1988] proposed improving efficiency by increasing the MVR in the direction of motion and similarly increasing mechanical advantage by increasing MMA in the direction of the applied force. Similar to the velocity and force ellipsoids used by Chiu [1988], MVR and MMA ellipsoids can be formed which indicate the best directions for increasing the MVR or the MMA.

Thus, based on the task at hand, joint configurations are selected that produce ellipsoids with desirable qualities along the given EEF trajectory. For instance, increased MVR may be desired for a process that needs to be completed quickly such as laser cutting, whereas increased MMA would be useful for a large force tasks such as grinding. In theory, we can extend the use of redundancies to reduce or increase some other property at the EEF, perhaps one related to grasping.

Redundancies can also be utilized to optimize other dexterity measures. Suggested measures include the Measure of Manipulability, the Jacobian's condition number, Joint Range Availability, and the minimum singular value [Klein, 1985]. For a given EEF location, a combination of these values can give a quantitative measure of the dexterity of a configuration.

Since the initial efforts discussed above, it is worth noting that there has been a substantial amount of work in the areas of configuration optimization for redundant system, both internal to the RRG [Tisius, 2009][Pholsiri, 2004][Pryor, 2002] and external [Ito, 2010][Lee, 2006][de M. Martins, 2006]. Patel [2005] provides a review for the interested reader. For the purposes of this effort, a system's parametric constraints, workspace geometry, and the above metrics provide a sufficient foundation upon which the complete the research objectives outlined in Section 1.2.

3.1.4.4 Representing Robot Capabilities

The work of Zacharias [2007] is most applicable to our research objectives. Zacharias developed the *capability map* to visualize a robot's kinematic capability in the workspace. These maps can identify locations in the workspace that are easy to reach and/or provide versatile manipulation. Furthermore, the maps provide a basis for finding robot arm target configurations that is useful for task, grasp, and motion planning. For

example, in the case of mobile systems, these maps assist in the placement of the manipulator base in order to complete a task.

Zacharias employed this new approach to overcome interpretation problems and inadequacies when using the manipulability ellipsoid and other dexterity measures. For instance, the *manipulability measure* [Yoshikawa, 1985] can be used to visualize singularities in the workspace, but the directional structure of the workspace, that is, the preferred Tool Center Point (TCP) orientations at points in the Cartesian workspace is not captured. Given a particular robot, there are certain regions in the workspace that can only be reached from specific directions (i.e. with specific TCP orientations), and this information is useful for subsequent task, grasp, and motion planning.

Zacharias' capability map representation involves a number of steps. First, the arm's theoretically attainable workspace is approximated (and overestimated) and then subdivided into equally sized smaller cube shaped regions. Then the configuration space of the manipulator is sampled according to a uniform distribution, and the position of the TCP is computed for each configuration using forward kinematics. The TCP position is mapped to the sub-cube in the workspace that contains this position. This produces a more accurate estimate of the actual reachable workspace.

Each cube is then inscribed with a sphere of diameter equal to the width of the cube. Next, N equally distributed points are generated on the sphere and a frame is attached to each of these points. The z-axis of the frame extends from the point to the center of the sphere, and then the frame is rotated about the z-axis according to a fixed step-size. Each of these frames represents a TCP frame to be tried by the arm. Inverse kinematics is then performed to determine if a TCP frame is valid/attainable. The *reachability index D* is calculated as the percentage of points on the sphere that have a valid inverse kinematics solution,

$$D = \frac{R}{N} \cdot 100, \tag{3-8}$$

where R is the number of valid inverse solutions, N is the total number of points on the sphere, and $R \leq N$.

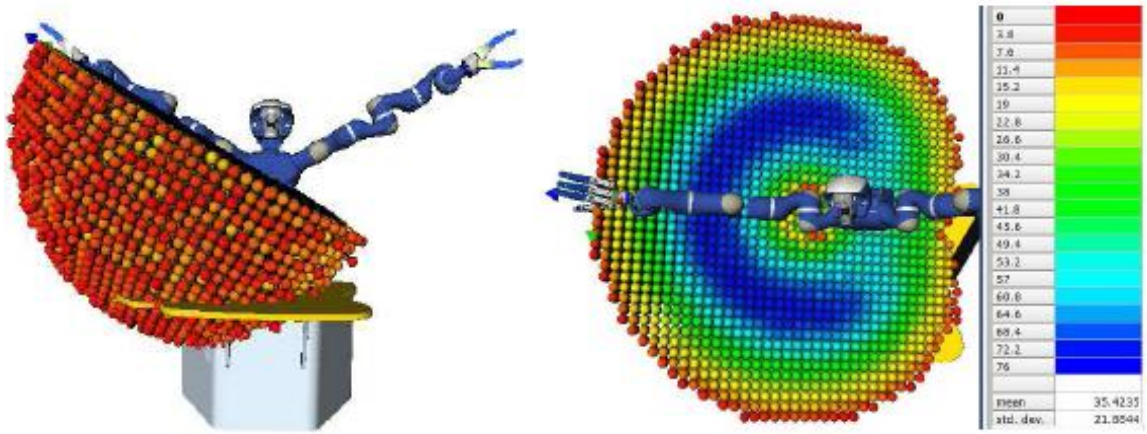


Figure 3-6: A cut across the workspace of the right arm and a planar section of the reachability spheres at the top of this cut. [Zacharias, 2007] Blue corresponds to the greatest reachability and red to the lowest.

The valid TCP directions are fitted to shape primitives such as cylinders and cones to represent the approximate volume that bounds the valid TCP frames. These primitives can be used for faster searching and testing for valid object approach directions and are also useful for data reduction purposes.

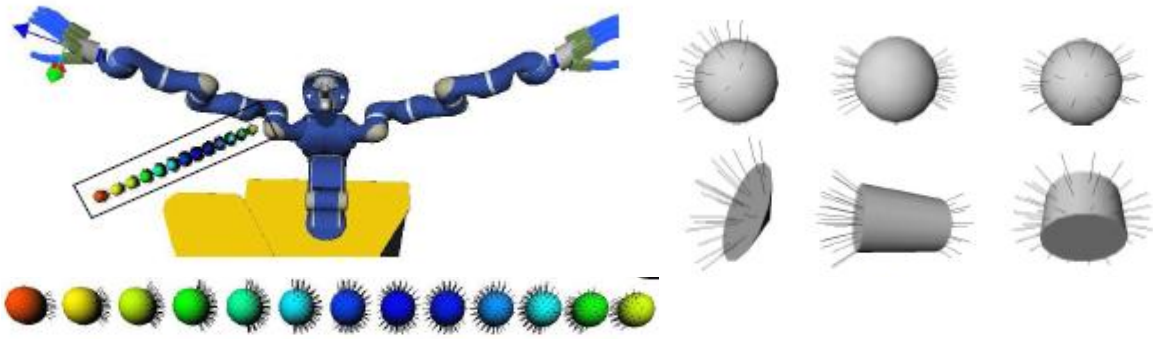


Figure 3-7: Capturing directional information at points in the workspace. Left: A sequence of reachability spheres showing lines where z axes of valid TCP frames exist. [Zacharias, 2007] Right: Shape primitives bounding the volume containing valid TCP frames.

Putting both components together results in the finished capability map of the workspace. An example is show in Figure 3-8. Regions with higher reachability indices correspond to areas of the workspace that have more versatile manipulation. Directional shape primitives can be used to find approach and grasp directions by choosing TCP frames that lie within the shape volume.

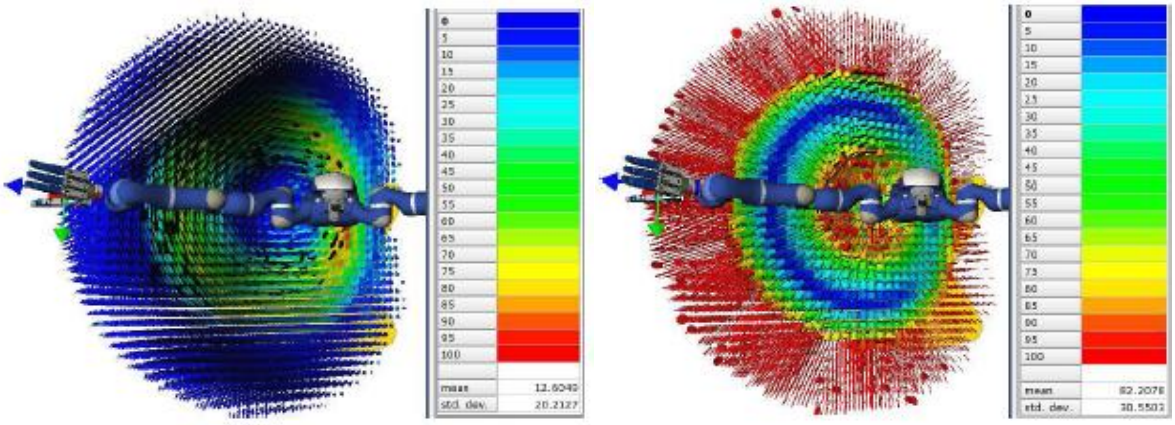


Figure 3-8: Capability maps with best shape fitting to the inverse kinematics data that replaces the reachability spheres. The left image contains cone best fitting and cylinder best fitting is used on the right. [Zacharias, 2007]

Building on this work, Zacharias [2008] used the capability map approach to position a mobile manipulator's workspace where a constrained linear trajectory could be performed. These linear trajectories are assumed to be known and motions occur in a *task plane* representing a planar subset of the capability map (Figure 3-9). In a task plane, all sampled TCP frames are the same for each sphere. A single TCP orientation can be selected to generate a *binary map*. In this workspace map, black regions occur where the TCP is attainable and white regions occur where it is not attainable. To conduct the motion plan, binary maps are generated for all possible TCP fames in the task plane. A pattern recognition technique is used to find regions in the binary maps where the given trajectory is possible.

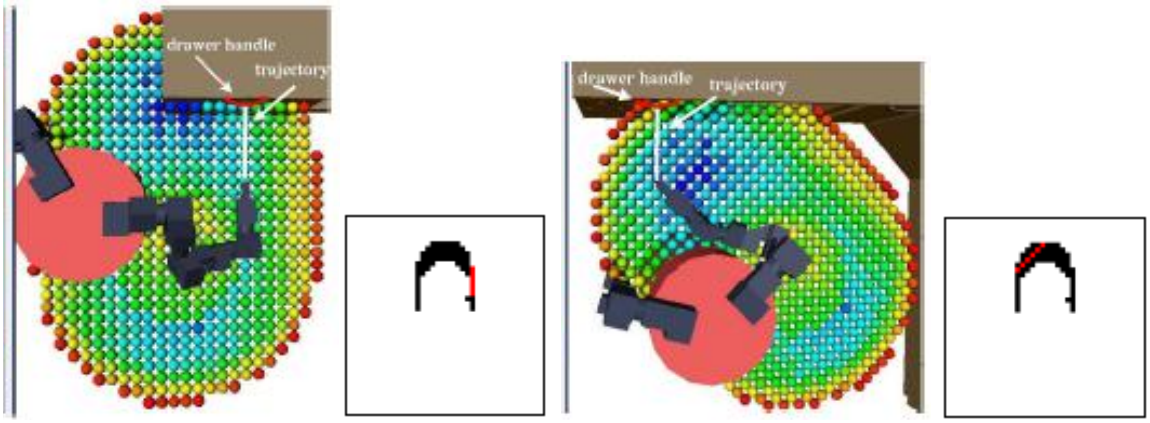


Figure 3-9: Performing a trajectory in a task plane for two different fixed EEF orientations. [Zacharias, 2008] The corresponding binary maps are next to each task plane, with the red lines indicating where the trajectory is being executed.

The feasible trajectories are then checked for consistency and collisions and then ranked according to criteria associated with improved positions based on the selected requirements. Based on 1) the starting point of the trajectory, 2) the task grasping point, and 3) the TCP frame, the mobile manipulator position necessary to execute the trajectory

can be computed. Zacharias [2008] found that the success rate of finding a mobile manipulator placement was ***increased from 2% to 70% by using the capability map approach instead of a brute force search of the workspace***. The trajectory search also identified the location in the workspace where valid trajectories were found the most.

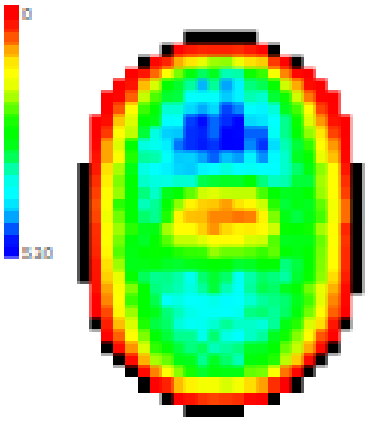


Figure 3-10: The locations of valid trajectories in the workspace using a pattern recognition search in binary maps. [Zacharias, 2007] Valid trajectories were found the most in blue regions, and black areas are reachable regions that are not used.

The example presented is most relevant to a mobile system where the search is for a platform base position that will allow for the execution of a particular task. This principle can be applied to task and layout analysis of fixed systems. Such a map could be utilized when positioning targets and fixtures in the workspace or when deciding where in the workspace a certain task should be performed. If multiple tasks are desired, different workspace configurations could be tried to determine the configuration that supports the completion of the most tasks. This work also could be extended to examine the viability of final grasp configurations with respect to initial grasp configurations as part of the objectives of our research.

3.1.5 Summary of Task Planning and Layout Analysis

We discussed a few but relevant approaches to the task and layout analysis of robotic systems. We also detailed some of the techniques for optimizing a manipulator's ability to execute multiple tasks in the workspace, something our research will also address. An accurate determination of a manipulator's grasping capabilities in the workspace and for each task will be important for reaching this goal, as well as developing a visual workspace representation to help generate a layout that can support a given set of tasks.

3.2 MODELING

The task and layout analysis may result in a feasible design for a robotic system, but in order for the actual physical system to operate, it must be accurately modeled. This includes modeling the manipulator, its surrounding environment, and obstacles. Manipulator modeling is well established, but modeling for collisions is still an active area of research. Other research areas (i.e. CAD and computer graphics) provide numerous examinations of the problem. Here we examine the tools available for accurately modeling the environment in a manipulator's workspace. Although research development in the area of modeling is outside the scope of this effort, our work needs to be compatible with the existing data collection and modeling techniques.

3.2.1 Modeling Objects

Modeling the environment requires a geometric representation of the real system, using CAD data, sensor-based visualization techniques, a collection of primitive shapes that approximate the object, or a combination of all of these. [Knoll and Tesar, 2007]

Sensor-based techniques will be addressed later. For now, we will review the use of CAD models and primitives to perform obstacle avoidance and collision detection. For well-structured environments with all objects known, CAD models are a good source of geometric data.

The RRG has developed the use of primitives to perform collision detection. [Perry and Tesar, 1995] These include spheres and cylispheres, where a sphere is given by a single point and a radius, and a cylisphere is given by two endpoints and a radius. They developed the mathematics for calculating the shortest distance between the surfaces of two primitives, where the witness points are the two closest points between the primitives. Superquadric surfaces were incorporated for modeling that required greater accuracy. Parameters can be varied to model such surfaces as parallelepipeds, cylinders, cones, bottles, funnels, vases, etc. These primitives were eventually built upon by the work of Harden and Tesar [1997] and Swint and Tesar [2004], but the general idea of using geometric modeling to perform collision detection remains the same. The second step after modeling using geometric objects is the ability to calculate the minimum distance between any two objects. Thus, given an adequate model, potential collisions can be detected.

Both collision detection and obstacle avoidance were implemented into DARPA's Trauma Pod project [Knoll and Tesar, 2007] to develop a mobile surgical workcell. Here, the only human in the workspace is the patient, with the surgical robot controlled remotely by the surgeon and other robots working autonomously. Such a workcell has a varied environment in which targets are generally fixed with respect to the manipulators, but the manipulators can move relative to one another. For most of the Trauma Pod components, CAD models were used for collision models. The modeling data was then

utilized to execute manipulator motion and monitor for potential collisions. [Spencer, 2008]

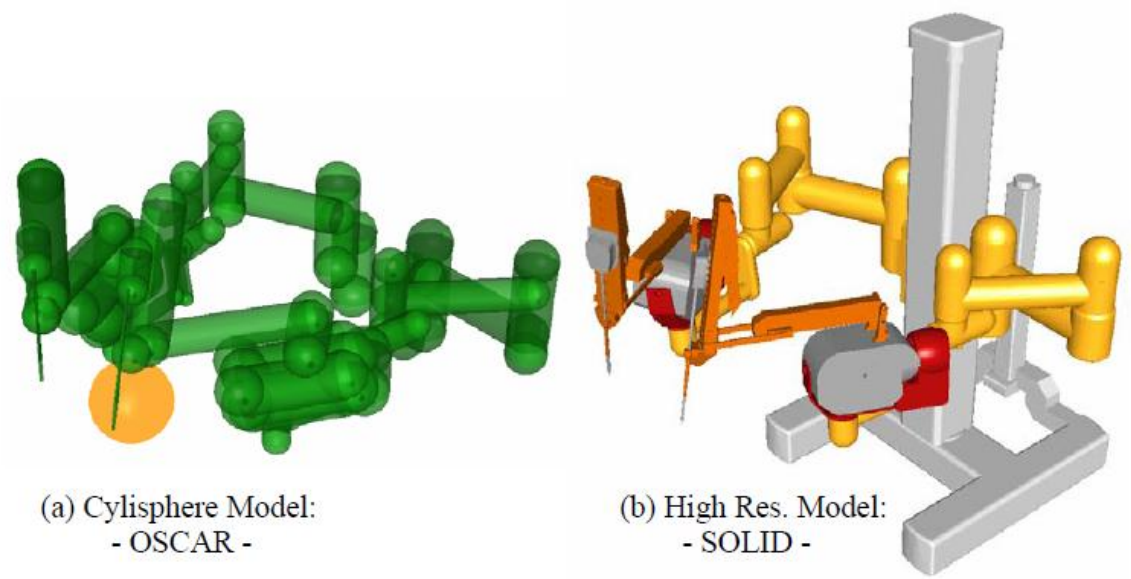


Figure 3-11: Models of Trauma Pod's da Vinci robot for performing collision detection [Knoll and Tesar, 2007] Model (*a*) models the da Vinci robot using cylisphere primitives. Model (*b*) uses SOftware Library for Interface Detection (SOLID) which models using simple primitive shapes and constructed shapes. [van den Bergen, 2003]

3.2.2 Vision Systems

Vision systems are another tool for developing a world model. Sensor data is often converted to polyhedral representations. In this way, vision systems can build a map of the manipulator workspace that is compatible with our existing obstacle avoidance techniques. In cases where the environment contains dynamic components other than the manipulators, vision can be used to generate an updated model. Vision systems can also be utilized to model unknown objects in the workspace that need to be manipulated. Dynamic modeling is particularly important for mobile manipulation. In

coordination with environments that have been modeled, vision systems are useful for checking the accuracy of the generated models. [Knoll and Tesar, 2007]

The RRG has successfully incorporated vision systems to ease the execution of pick-and-place tasks. [Hulse, 2009] Sensor data from a Swiss Ranger (SR3000) is gathered and processed to create a simplified user-interface for grasping in uncertain environments. This data is then used to generate a 3D point cloud which is overlaid on a video feed. This allows the operator to select an object to grasp on the video image rather than using a joystick or other manual controller to command the EEF. This capability was also demonstrated with a Bumblebee II stereo vision system. [O'Neil, 2010]

Figure 3-12: A video interface for the selection of objects to grasp that uses data generated by a Swiss Ranger. [O'Neil, 2010]

3.2.3 Summary of Modeling

This section briefly covered basic principles behind environmental modeling, with an emphasis on modeling obstacles and collecting modeling data with sensors. As we will see, model data plays an important role in analyzing tasks and planning motions.

Without an accurate and comprehensive world model, there is little hope that a manipulator will effectively carry out tasks without unwanted contact with the environment. Although we do not seek contributions to the modeling area, it is important to mention that we assume we have a means of obtaining an accurate world model when applying our work.

3.3 MOTION PLANNING

There are different factors that must be considered when planning the movement of a manipulator in its workspace. Path planning involves just the geometric (kinematic) specifications of position and orientation. [Achint and Tesar, 2009] Trajectory planning adds another level of planning that includes linear and angular velocities. Motion planning involves the incorporation of system dynamics into the trajectory generation process.

Additionally, there are different planning techniques relative to the task, such as gross and fine motion planning. [Hwang and Ahuja, 1992] Gross or global planning applies to motion in free space that is larger than objects' sizes, where the positional error of the robot is monitored to ensure that unexpected collisions do not occur while executing collision-free paths generated by a planner. Fine or local planning deals with moving objects through a narrow space that requires motion accuracy that exceeds the robot's positional accuracy. The emphasis of this effort is on gross path planning and geometric specification. Although local planning is often associated with grasping, our impetus is not on grasp planning. However, we will touch on this type of planning in Section 3.4 when we present grasping.

Path planning and the modeling of objects in the environment are closely linked. In the case of a pick-and-place task, the initial and final grasps and the path between them

all are subject to environmental constraints. For instance, consider the flow of the motion planner used for Trauma Pod in Figure 3-13.

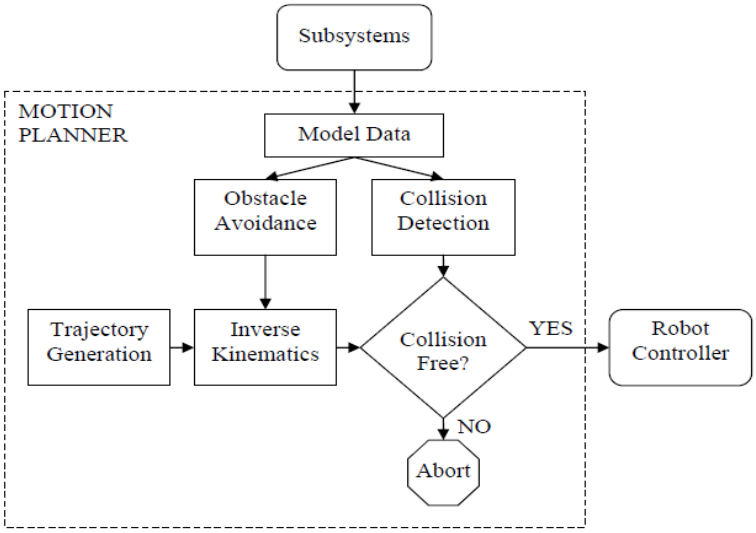


Figure 3-13: Structure of the motion planner used for Trauma-Pod [Knoll and Tesar, 2007]

This planner has five components: trajectory generations, forward and inverse kinematics, obstacle avoidance, collision detection, and model data updating. The model data plays a central role in the motion planning of the system, contributing to obstacle avoidance and collision detection. In regard to the environment, motion planning can be static or dynamic. [Hwang and Ahuja, 1992] For the static case, all information about the obstacles is known a priori. In dynamic planning, the robot makes a plan based on the available obstacle information and learns more about the obstacles as it moves, re-planning its path accordingly. In our work, we would like to handle both static and dynamic obstacles.

3.3.1 Types of Trajectories

As shown in Figure 3-13, an essential component of the motion planner is the trajectory generator. The two main forms of trajectory generation are joint space planning and EEF or Cartesian space planning. In joint space planning, each joint is treated like a one DOF system, and a full trajectory is computed individually for each joint. Common methods for generating smooth single joint trajectories can be found in [Rajan, 2001]. Joint space planning does not allow for EEF control throughout the motion.

Cartesian space planning, however, is utilized to generate EEF trajectories. Since joint angles must be passed to the actual robotic system, inverse kinematics must transform the desired end-effector position and orientation to joint angles. Cartesian space trajectory planning is especially useful for tasks that require the EEF to be constrained to certain motions. For example, surface polishing or moving an object that must remain face-up throughout its entire movement. Techniques for generating Cartesian space trajectories/curves can be found in March and Tesar [2004]. Although system dynamics must be considered to produce smooth and accurate motions, we will not discuss them here.

3.3.2 Criteria Based Motion Planning

Computed trajectories can be compared using criteria. Tesar [1976] formulated five motion curve criteria to help compare the quality of different motion profiles for 1 DOF systems. The spatial curves produced by Cartesian space planning can also be analyzed using criteria based on such curve properties as curvature and torsion. [March and Tesar, 2004]

When executing obstacle avoidance, although collisions are avoided, the paths generated may contain undesirable properties such as sharp corners or, more generally and intuitively, motions that would cause concern for an operator supervising the system. March and Tesar developed a method of EEF motion planning that used curve criteria in the decision-making process in addition to joint level criteria. His method was as follows:

1) A new EEF position is determined using path planning criteria
2) A set of joint options is perturbed about this new EEF position
3) The best joint position is determined based on joint-level performance criteria

The system does not have to be redundant for this method to be useful. If one or more of the EEF coordinates is unconstrained, then multiple solutions exist. Thus, the orientation of the EEF could be perturbed to generate a set of joint options. These joint solutions could be evaluated based on obstacle avoidance or other measures such as MOT and JRA. A similar process could be used to evaluate the base location of a manipulator if it is variable.

3.3.3 Obstacle Avoidance

Obstacle avoidance is associated with path planning but is performed in real-time on redundant manipulators and often comes down to a resource allocation problem. Typically, methods incorporate criteria where a joint solution set is generated using inverse kinematics then ranked using criteria. If obstacle avoidance is desired, this ranking process can fuse other metrics with obstacle avoidance criteria.

Groups of obstacle avoidance algorithms include heuristic, kinematics, free space modeling, potential function, and nonlinear optimization methods. [Sreedhar and Tesar, 1990] The RRG has focused its work on the Artificial Potential Field (APF) method. Harden and Tesar [1997] looked at using APFs and performance criteria to execute real-time obstacle avoidance. In this technique, each location in the manipulator's workspace has some potential associated with it. Obstacles have higher potentials associated with them, with increasing potential as the object is approached, and target locations have the lowest potentials. The goal is "to position a manipulator in the workspace such that the overall potential encountered by the manipulator is as low as possible while still accomplishing the desired task." [Harden and Tesar, 1997] The APF can be mapped to artificial joint torques and integrated into a motion planning algorithm. Obstacles are avoided by minimizing the potential encountered by the manipulator as it moves through the workspace. Harden and Tesar used obstacle avoidance criteria based on minimum distance calculations between the workspace modeling primitives and also artificial potential and artificial forces.

The magnitude of the potential is often based on the minimum distance between a manipulator link and an obstacle, so the modeling and mathematics for distance calculations between primitive shapes becomes important. This method of obstacle avoidance uses criteria-based decision-making where the criteria are based on either the calculated minimum distance or the APF between a manipulator link and an obstacle. For the APF criteria, the artificial forces on the manipulator are used to calculate the artificial torques about each joint. Once these torques are determined, they can be transformed to get the equivalent forces/torques at the EEF. The resulting force vector indicates which direction the EEF should be moved to best avoid obstacles while the direction of the torque vector indicates the best axis to rotate the EEF about to increase

the distance between itself and obstacles. These vectors also give information about the proximity of the manipulator to obstacles. Minimizing the force and torque vectors are the simplest criteria.

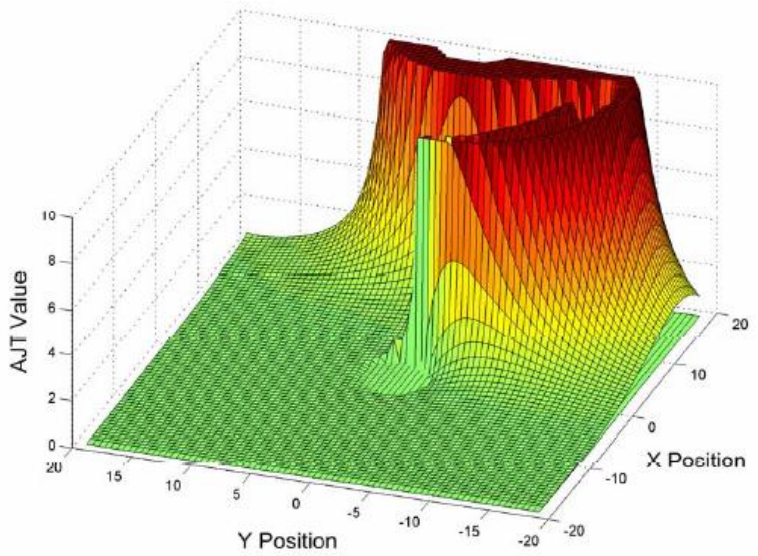


Figure 3-14: A plot of the Artificial Joint Torque (AJT) criterion for a 2 DOF planar manipulator with one obstacle. [Spencer and Tesar, 2007]

3.3.4 Configuration Space

Lozano-Perez [1989] considered global motion planning using the Configuration Space or C-space. They defined a configuration as "any set of parameters that uniquely specify the position of every part of a system," and the C-space is the space defined by those parameters. C-space obstacles are a sub-group of configurations that cause collisions.

Lozano-Perez [1989] presented the example of a 2 DOF system that contained obstacles in its environment, as seen in Figure 3-15.

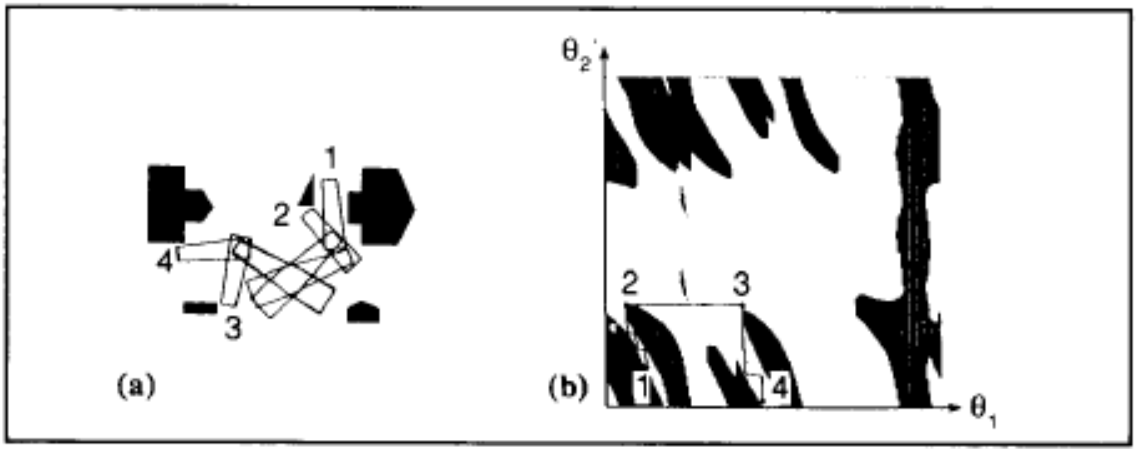


Figure 3-15: The configuration space for a 2 link planar system [Lozano-Perez, 1989]

The C-space has Joint 1 (θ_1) on the horizontal axis and Joint 2 (θ_2) on the vertical axis. Configurations correspond to a pair of θ_1 and θ_2 values. The dark regions denote configurations that collide with obstacles. Motion planning in the C-space is just a matter of finding a connected path from the start position to the goal position that does not touch any collision regions. In this C-space, because the parameters are angles, the right side actually connects to the left side and the top connects to the bottom, forming a C-space that is actually the surface of a torus.

The dimension of a robot C-space is the system's DOF. Forming a six-dimensional C-space for a 6 DOF system is computationally expensive and cannot be visualized. Instead, Lozano-Perez [1989] used some heuristics to improve computation time. The first simplification uses a local planner with a modified APF approach for small grasping motions near the initial and goal positions. This approach will be covered in more depth in Section 3.4.1 due to its relevance to grasping. Large motions were limited to the first three joints, but the last three joints could still be changed if desired.

The approximation of obstacles and the arm was also modified to simplify the computation of the C-space obstacles.

Essentially, the developed motion planner generated three three-dimensional slices of the underlying six-dimensional C-space. C-space obstacle construction, however, is limited to the first three joint angles, so the slices are built for a particular span of the first three joint angles. The first slice is built with the wrist angles fixed at their value at the start of the path. A second slice is built with the wrist angles fixed at their values at the end of the path. The last slice is built for the range of wrist angles between the start and end of the path. Free-space representation in these three slices is then linked into a single free-space representation that is searched for a path. Thus, this global approach can be utilized to take care of motion planning for gross motions.

Many motion planning techniques used today incorporate the C-space. Hwang [1992] defined the basic issues and steps of any motion planning formulation: C-Space construction, object sensing and representation, motion planning approach, search method for finding feasible plans, and local optimization of motion. We have already described methods for C-space formulation, obstacle representation and sensing, and some approaches to motion planning.

Of the many approaches to motion planning, including the APF approach already discussed, two common techniques are the skeleton and cell decomposition. Essentially, these methods provide a way of describing the free space, with each feasible node or cell representing a robot configuration. The skeleton approach is also referred to as the retraction, roadmap, or highway approach. [Hwang, 1992] The free-space of the manipulator's C-space is reduced or mapped to a network of one-dimensional lines representing valid paths called the skeleton. The initial and goal positions are also

connected to this network. A graph searching algorithm is then used on this skeleton to find a series of connected lines from the initial position to the goal position.

There are various types of skeletons that can be generated for this method, and some common methods randomly sample the C-space. In the Probabilistic Road Map (PRM) sampling method, [Kavraki, 1996] configurations are randomly generated, creating nodes or a graph. Then a local path planner determines if a path exists between any two configurations, generating an undirected graph. The Rapidly-Exploring Random Tree (RRT) is a variant of the PRM method. [LaValle, 1998] This algorithm grows a search tree out from the initial configuration and uses biased sampling to sample unexplored regions.

In the cell decomposition method, the C-space is decomposed into a set of smaller cells that may or may not be the same size or shape. Adjacency relationships are defined that determine whether or not transitions are allowed between any two adjacent cells. In the case of obstacles, the obstacle boundaries can be used to generate the cell boundaries, or the C-spare can be divided into a grid of similar cells and each cell is tested for being inside or outside an obstacle. A path is found by searching for a connected series of non-obstacle cells between the initial and goal cells.

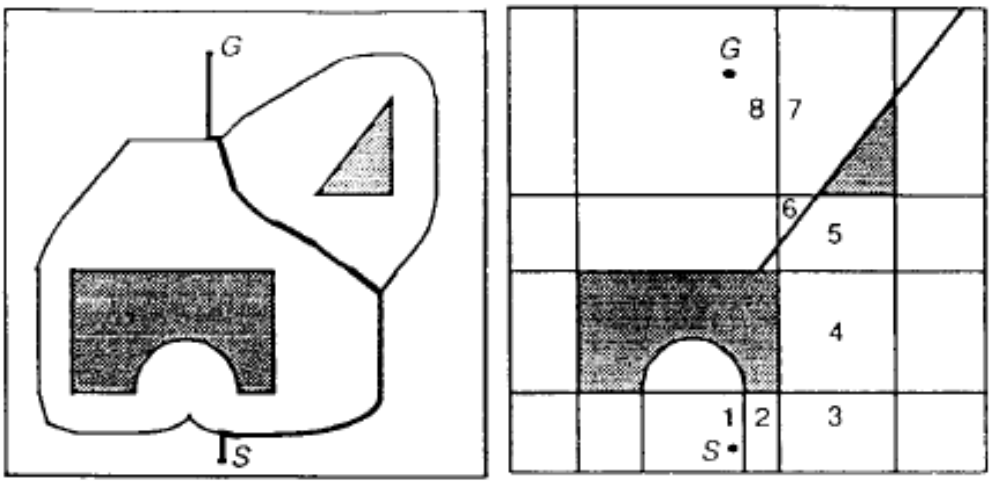
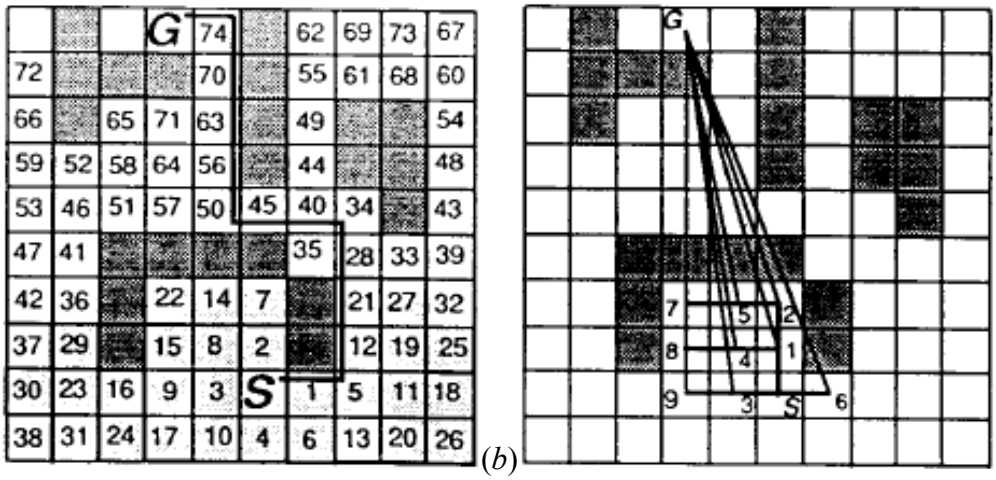


Figure 3-16: An illustration of the skeleton and cell decomposition approaches. [Hwang, 1992] Left: A skeleton representation with two obstacles (shaded) and starting point S and goal point G. Right: Cell decomposition method using the same obstacles. The path found traverses eight cells labeled from 1 to 8.

Once a method has been used to describe the free space, the space is searched for a valid path given a start and goal configuration. The search algorithms for cell decomposition and skeleton methods are well defined in Hwang [1992]. Some of the more common search algorithms are depth-first, breadth-first, best-first, Dijkstra, and *A (pronounced "A-star"). In the depth-first search, the search expansion is governed by predetermined, preferred search directions from the current configuration. In this case, the solution found is not always the shortest. The breath-first search can find the shortest path, but it will examine a large part of the space. The depth-first and breadth-first algorithms are also not very efficient. The best-first search takes into consideration the distance of a configuration to the goal, and uses this information to make a move to an adjacent configuration that is the shortest option to the goal. In some cases, this algorithm takes a long time to reach the goal.



(*a*) (*b*)

Figure 3-17: Examples of two search algorithms. [Hwang, 1992] Image (*a*) shows a path from start S to point G using a breath-first search. Image (*b*) demonstrates the use of a heuristic in the *A algorithm. Here, the heuristic is the straight-line distance from a prospective configuration to the goal, and this heuristic helps drive the search.

The *A algorithm also considers the distance of a configuration to the goal and is good for finding the shortest path. To do so, the algorithm needs a heuristic that underestimates the cost from the current configuration to the goal. This is often the straight line distance between the configurations. The cost is then the sum of the actual cost of the move from the start configuration to the current configuration and the heuristic cost estimate. The *A algorithm is a special case of Dijkstra's algorithm (the latter does not have a heuristic). Hwang [1992] presents some guidelines for selecting among these various search algorithms.

These types of motion planning approaches are useful for our work. We are going to use an approach similar to the capability maps described in Section 3.1.4.4. We will not randomly sample the C-space, but our maps will consist of a discretized grid of attainable points in space that contain data for which relationships can be found between adjacent grid points. Unreachable points in the grid will be determined using the

collision model and reachability, and cost estimates will be derived from the dexterity at a grid point. We are interested in finding any valid path, but we would like to find a direct path. The *A algorithm is a best-first algorithm that finds a path if it exists and also has the added benefit of finding the lowest cost path. Thus, in our approach to motion planning, we will use capability maps to generate a connected grid/graph that can be searched for valid paths using the *A algorithm.

3.3.5 Summary of Motion Planning

Motion planning is closely tied with system modeling. For any real motions, the system must be aware of potential obstacles and plan its motions accordingly. In order to choose optimal grasping, environmental obstacles must be accurately modeled to determine which grasps are possible and if a collision-free path can be constructed between the chosen grasps. Our work will build upon the motion planner structure, using dexterity criteria to guide motions. In the case of grasp configuration planning, such criteria could be incorporated within the same redundancy framework as obstacle avoidance. For information on other planners, Achint and Tesar [2009] provide an excellent overview of several common motion planners.

3.4 GRASPING

Much of the previous work in the RRG deals with fixed grasping. Movement is commonly between fixed/stored EEF frames. Furthermore, it is assumed that the geometries and initial and final object locations of objects to be grasped are known in order to hard-code both the grasp and the grasp approach direction.

On the other hand, we are searching for a method to select valid grasps in dynamic environments when tasks are not well-defined. Specifically, we are more concerned with grasps as they apply to the location of the EEF tool point and the resulting kinematic state of the manipulator. Therefore, we will review some past efforts of researchers in developing grasping strategies for non-static cases. We will also see that this work is intertwined with the areas of task planning, modeling, and motion planning.

3.4.1 Local Motion Planning

As discussed, a distinction is often made between local motion planning for fine movements and global motion planning for gross manipulator movements. Local planning is often used for grasping since these movements commonly take place in proximity to other objects and precise movement is necessary around the object to be grasped. In this context, Lozano-Perez [1989] considered grasps for the initial and final positions of a task. They explored how the environment and objects around the initial and target positions affected the choice of a grasp. In regions with obstacles, the choice of grasp must be analyzed to determine its effect on finding a path to the goal. Lozano-Perez considered using only "safe" initial and final positions such that paths between the grasps are available and collision free. These paths were determined using C-space/global motion planning (in Section 3.3.4) once a grasp was chosen.

This left the task of determining which grasps to use that also avoid collisions. The general idea was to characterize the reachable grasps at the initial position then characterize the collision free grasps at the target position. The grasps in both of these two sets are those grasps which are reachable at the initial position and collision free at both positions. If valid grasps at both locations could not be found, then regrasping was

used (see Section 3.4.3). A second method for determining grasps, which was not used and is seen in Figure 3-18, is as follows:

> "Compute the transformation T that maps the grasped part from its initial to its final location. Apply the inverse transformation T^{-1} to a copy of the obstacles near the final position of the part. Add these transformed obstacles to the obstacles near the initial position of the part. Find a path to any legal grasp that avoids both sets of obstacles."

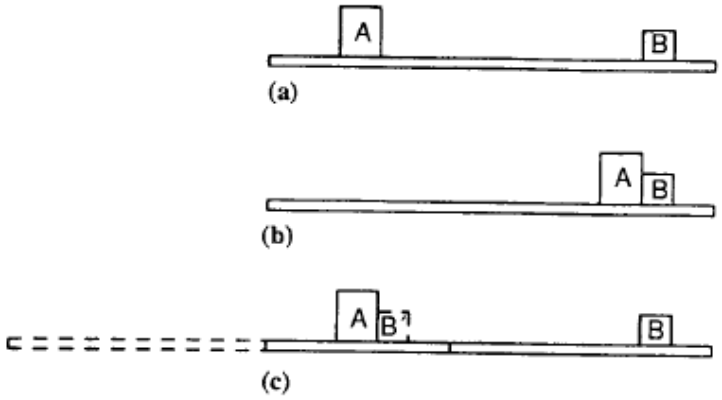


Figure 3-18: A 2nd formulation for determining valid grasps. [Lozano-Perez, 1989] (*a*) The grasped part A and obstacle B by A's desired final position. (*b*) Mapping A to its final position. (*c*) Applying an inverse transformation to map A and B back to the initial position.

The motion planning for the pick-and-place tasks was a combination of local and global motion planning. Local motion planning was used for motions when the object is grasped. To avoid obstacles and guide the gripper to the object grasped, a "quasipotential" method was employed which is derived from the APF method (see Section 3.3.3). Surrounding the gripper at a distance *d* are *bump lines* (Figure 3-19). If the gripper-to-obstacle distance is greater than *d*, then no force is felt, but it is less than *d*, the force is high enough to prohibit motion toward the obstacle.

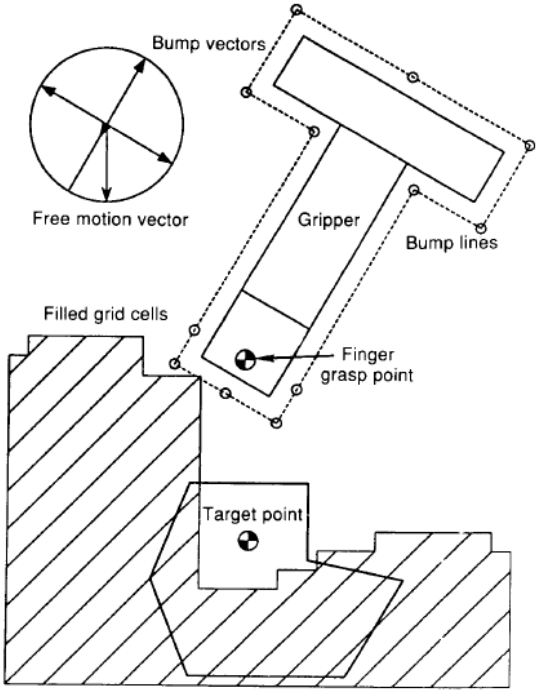


Figure 3-19: Schematic of the local motion planning used by Lozano-Perez [1989] when grasping an object.

If the gripper must remain in the plane parallel to the object grasp faces, then motion in Cartesian space now has 3 DOF: x, y, and θ. *Bump vectors* are defined as unit vectors perpendicular to bump lines, and they point away from the gripper. The gripper's motion is along the *goal vector* defined as the vector connecting the finger contact point to the object grasp point. Motion in the plane is found by traversing a grid with obstacles corresponding to filled grid cells. The gripper moves along the goal vector, and the bump lines are checked against filled grid cells for collisions. If a collision occurs, the bump vector corresponding to the bump line goes to zero length. A unit circle is constructed that maps the goal vector and the bump vectors. If the goal vector does not have a component in the direction of a zero-length bump vector, then the gripper continues to moves in the goal vector direction. Otherwise, the gripper moves along the nonzero

bump vector that is closest in direction to the current goal vector. Collisions with bump lines also produce a torque which rotates the gripper a fixed amount about the finger grasp point. The path is found when the fingers sufficiently overlap the target point. In this way, an approach for a grasp can be planned.

3.4.2 Heuristic Grasping and Obstacles

Zacharias [2006] highlights the links between task planning, path planning, and grasping. In order to plan a geometric trajectory, a collision free and reachable start and goal robot arm configuration is needed. They present a grasping method that chooses "where to grasp an object depending on the robot's kinematic structure, surrounding obstacles and the position of the robot relative to the target object." They used a heuristic method to determine the search area for grasp approaches, noting that humans tend to grasp objects only in a certain area with respect to their own position and orientation.

Two methods are proposed to analyze where preferred grasping directions are located around an area containing an object. In the brute force approach, points are sampled around the circumference of an object and inverse kinematics is computed for each sampled point/position. Then good grasp approach directions are searched for in the largest connected region of valid inverse solutions around the object's circumference.

The second is a heuristic approach based on the observation that humans often make grasps in a certain area about an object with respect to themselves. Thus, the grasp direction in the object's coordinate system depends on the robot's position. The y-axis of the object's coordinate system is defined by the line extending from the position of the robot arm base to the center of the object, and the z-axis points up. Valid grasps are searched for in the fourth quadrant in the x-y plane (Figure 3-20).

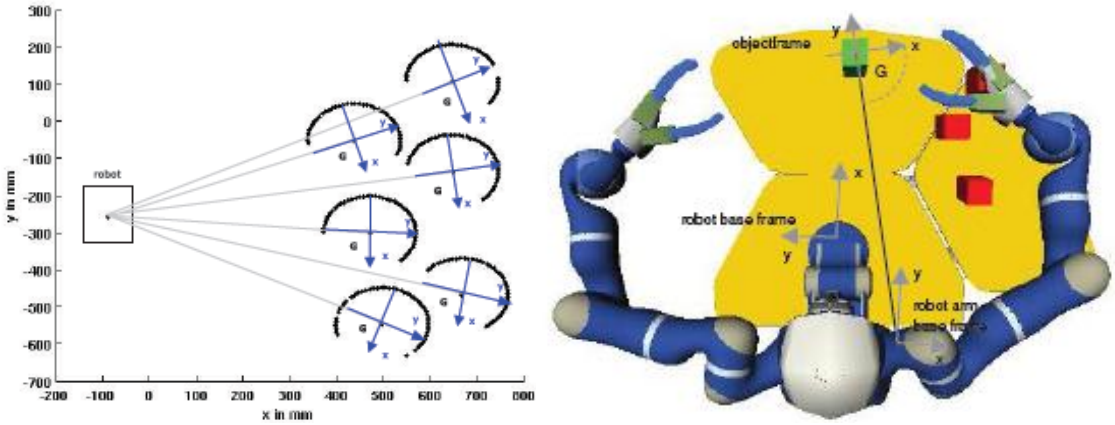


Figure 3-20: Illustrations of two methods for determining preferred grasping directions. Zacharias [2006] Left: The reachability of six objects is analyzed using the brute force method. Right: The heuristic approach showing the derivation of the preferred grasping area *G* about the object frame.

The heuristic approach always searches in part of the largest connected stretch of valid inverse kinematic solutions found by the brute force method. However, the heuristic approach does not identify all the solutions found using brute force. In most cases, these lost positions are also farther away from the robot and not as desirable. Additionally, the brute force method does not scale to 3-D, making the heuristic approach slightly more favorable overall. A potential field approach was also considered, but presents more complexity than needed and sometimes has problems with small clearance between obstacles. [Borenstein, 1991]

The grasping problem becomes more complex when obstacles and collisions are considered, especially when the target object is completely blocked by other obstacles. Zacharias [2006] adapts the vector field histogram method of Borenstein [1991], centering and orienting a polar histogram on the target object's coordinate system and sampling sectors in the fourth quadrant defined by the heuristic approach (Figure 3-20). They compute the obstacle density values for a sector by introducing the concept of

obstacle influence rather than by using functions for density computation. A grasping preference of the target object is also determined. The influence values of the target grasp preference and all obstacles are used to determine the accessibility of a sector and the particular approach direction it represents. The approach direction corresponding to the sector with the minimum total influence value is chosen. This value is compared to a threshold value to determine whether the target is blocked by an obstacle.

If the target is blocked, some *rearrangement planning* [Ben-Shahar, 1998] is necessary. Obstacles are removed from around the target by considering the influence of object removals on the target *graspability* and using an algorithm that builds a tree to find a sequence of valid obstacle removals. Object accessibility is determined using the process above, and the tree structure is then searched to find the optimal removal sequence.

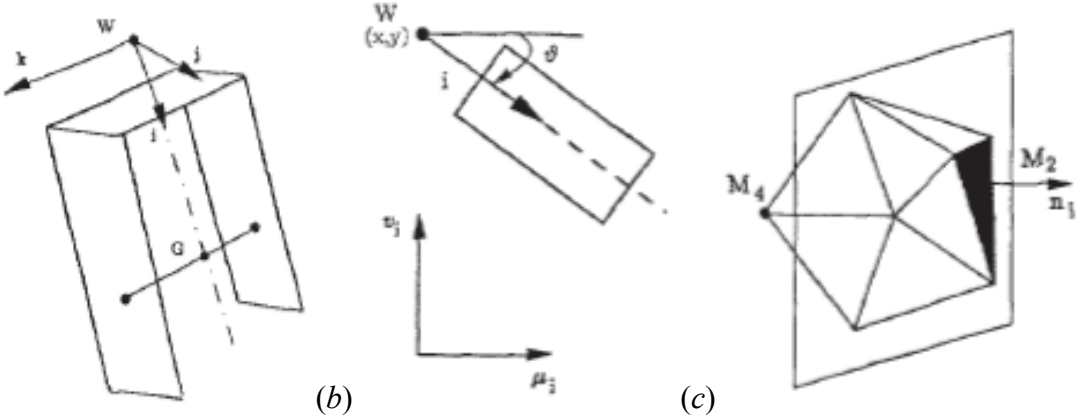
3.4.3 Regrasping

Tournassoud [1987] investigated regrasping an object whenever the robot's grasp on an object was not compatible with the task to be performed. They defined the overall problem as "given an initial and a final grasp of an object, construct a sequence of ungrasping and grasping operations connecting them." They identified a few main constraints that lead to the need for regrasping:

1) Geometric interactions between the object, the manipulator's hand, and the environment
2) The kinematics of the manipulator, in particular, the mechanical limits on the joints
3) The inability to find a collision-free path to complete the entire task

Regrasping operations can be divided into two groups. The first includes motions during which the object remains in contact with the hand, which can only be performed by a dexterous-hand capable of precise force control. The second class excludes all dynamic motions and the geometric relationship between the object and the hand is fixed during moves. In this case, changes involve setting the object down, finding a different grasp, then picking up the object again. Tournassoud limited the discussion to the second class of regrasping which used parallel-jaw grippers and not dexterous hands.

Objects were modeled as three dimensional polyhedron and stable grasps were determined based on finger contact points on the polyhedron. During a stable grasp, the fingers will lie in a plane. From this plane, the grasp point is selected, which is defined as a point bound to the hand lying midway between the surfaces of contact on the fingers and located near the tips of the finger. This point is used to establish a frame attached to the grasp. A parameter θ is defined as the angle of the fingers in the grasp plane in a frame bound to the object. These are the parameters used to define a grasp and also search for a grasp in the *Grasp Space*.



(*a*) (*b*) (*c*)

Figure 3-21: Regrasping formulations. [Tournassoud, 1987] (*a*) Model of the gripper, where *W* represents the wrist point and *G* is the grasp point. (*b*) Positioning of the gripper in the task frame *(x,y)*, and the orientation of the fingers θ. (*c*) The grasp plane with normal n_1 centered on the object. For the grasp, M_2 is a contact plane and M_4 is a contact point on the object.

Candidates must also be found for the placement of the object on the table for regrasping. The *Placement Space* is dependent upon the normal to the object face that makes contact with the surface, which implies a specific location for the projection of the object's center of mass on the table and the angle of rotation of the object around a vertical axis. The best points are those that lie in the robot's dexterous workspace where multiple solutions for the inverse kinematic exist.

A grasping operation is then a choice of some combination of the Grasp Space and Placement Space, basically a transition between two grasps that utilizes a placement that is compatible with both grasps. To do this, these spaces are combined to form the *Grasp-Placement Space*. The subset of this space that is the set of all legal conjunctions of grasps and placements is of particular use when choosing grasps. There are a few more considerations when selecting from the Grasp-Placement Space. One of these includes computing the C-space obstacles or finding θ values that cause the hand and object to overlap. Another is finding a solution to the inverse kinematics that does not violate joint limits.

3.4.4 Grasping With Complex Hands

Grasp planning becomes more complex as the type of gripper/hand used becomes more complex. These dexterous hands are more versatile, allowing them to grasp objects of various shapes and perform fine manipulation. Much complexity stems from the coordination of the hand fingers which can each be modeled as an individual manipulator. For these hands, Hester and Tesar [1998] broke the process of obtaining a grasp on an object into four phases:

1) **Object acquisition** – determining the type and position of the target object
2) **Grasp synthesis** – determining the appropriate hand position and configuration for type of object and task
3) **Approach** – positioning the hand near the object
4) **Grasp execution** – obtaining the actual grasp of the object

Hester and Tesar [1998] focused on the grasp synthesis phase for a dexterous hand, which "is essentially configuration management for a robotic hand." For grasp synthesis, a "good grasp" is determined based on five properties. [Shimoga, 1996] The first three properties are required for a successful grasp, and the last two provide a measure of the grasp's capability to perform certain tasks:

1) **Force closure** – the ability of a grasp to resist an arbitrary external force
2) **Equilibrium** – the ability to maintain a grasp on an object
3) **Stability** – the ability to resist external disturbances
4) **Dexterity** – the ability to impart motion to the grasped object
5) **Dynamic response** – the compliance of the grasp

Hester and Tesar [1998] looked at criteria-based decision-making in regard to grasp planning for the precision grasp of an object with a multi-fingered hand based on force and motion related task requirements. The complexity of this synthesis is reduced by evaluating an individual grasp based on force closure analysis and the calculation of force and motion related to grasp criteria. A two-phase routine was used to first select an initial grasp based on finger-level criteria then optimize the grasp using hand-level

criteria. In this case, the operator is responsible for the hand's approach to the object. Manipulation after obtaining an initial grasp is not addressed.

The preliminary grasp configuration consists of the best collision-free pose for each finger based on finger criteria which included the velocity and force transmission ratios. [Chiu, 1988] In the optimization process, the preliminary grasp configuration is the starting configuration for a direct-search conducted on the grasp configuration space using hand level criteria. This includes a "grasp quality factor" [Ferrari, 1992] that evaluates the ability of contact forces to resist external loads. Finally, the final grasp configuration found is evaluated for equilibrium and stability.

3.4.5 Summary of Grasping

The above approaches consider previously discussed topics such as obstacle modeling and path planning. Developing new grasping strategies will require knowledge in these and other areas. In addition, compliant grasping techniques [Dollar, 2005 and 2010][Xiong, 2005] are addressing issues associated with grasping forces and model uncertainty. Addressing these issues are active areas of research at The University of Texas. For a deeper overview of research involving complex hands, see Shimoga [1996].

In our research, we will focus on the manipulator's configuration for grasping and approach phases. Thus, we will only consider parallel jaw grippers or simply the Cartesian orientation of the hand in order to better decouple the manipulator's grasping strategy from that of the gripper's internal configuration. Yet the methods here should easily integrate with those determining the best approach to grasping. Assuming that such grasp configurations are identified, the set of possible gripper configurations can also be mapped to manipulator configurations using the methods developed and presented below.

Chapter 4 **Research Approach**

Chapters 2 and 3 survey previous robotics research related to our work. The research most relevant to our objectives is that of Zacharias [2006, 2007, and 2008]. Our first objective is to construct a similar graphic representation of the workspace that accurately captures a robot's kinematic capabilities. This representation reveals workspace regions where more valid grasping configurations are located (and manipulability is increased). Then we apply the information in the workspace map to system design and operation. For operation, we develop a grasping strategy that seeks to effectively *use valid configurations in the workspace to make more informed and better grasping and motion planning decisions*. This will improve system functionality and aid the operator in the control of the system.

4.1 APPROACH JUSTIFICATION AND BENEFITS

Other techniques, like Chiu [1987], Dubey [1988], and Abdel-Malek [2004] use mathematical formulations to determine and utilize dexterity. Such formulations can sometimes lack an easy connection to the physical properties of the system. Chiu presents the velocity and force ellipsoids to help visualize properties of a single configuration. However, the work of Zacharias [2007] presents visual tools that better connect physical manipulator configurations to the workspace. In the latter approach, the quality of the reachable workspace is known, not just the size/boundary. The capability map approach also easily extends from non-redundant to redundant systems. Zacharias' sampling approach for valid configurations throughout the workspace provides information about dexterity using a brute force approach, but it is exactly the information an operator needs to know and can intuitively apply to design and operations.

Such a tool formalizes the system layout analysis, eliminating time-consuming trial-and-error methods. We can use a map to locate task targets and trajectories near areas of high manipulability. This is a more intuitive way to determine the best base location for manipulating targets than those incorporating mathematical formulations (i.e. the method proposed by Abdel-Malek [2004]). Positioning based on a manipulability measure improves system flexibility by allowing the robot to grasp and move targets in multiple ways. Furthermore, the maps can also be utilized in manipulator design and selection. This can be accomplished through comparison of maps generated for the same manipulator with different dimensions or for completely different manipulators. A more qualified robot for a process means greater efficiency and capability.

The design stage develops the system configuration prior to operation, but information about the workspace is also useful once the system is operating. Increased system autonomy decreases operator responsibility, but the operator still needs to stay informed of system performance. This is particularly critical for systems in dynamic environments. Although system design may be optimized for the initial tasks and workspace configurations, there needs to be a way to complete these tasks *and new tasks* after changes to the environment or mission. An updated capability map can help the system or operator determine how to complete new tasks.

Zacharias [2008] also used capability maps to perform constrained linear trajectories. Similarly, another goal is to use this map as a motion planning tool. This tool can be particularly useful for path planning in dynamic environments. Essentially, the system can decide based on the initial and final grasp locations and manipulability information stored in the map whether it can find grasping configurations and a path that allows a task to be executed. Thus, the capability map is used to expedite Cartesian space motion planning for movements between frames and along paths that are not predefined,

increasing system autonomy. **In general, the capability map helps us devise a grasping strategy based on manipulability that selects the grasping configurations and trajectories needed for the completion of new/unknown tasks.**

4.2 THE CAPABILITY MAP APPROACH

Our approach is based on the steps Zacharias [2007] used to develop capability maps, which we will refer to as the *Capability Map Approach* (CMA). The CMA includes workspace discretization, randomized sampling of the C-space, determining reachability using inverse kinematics, and capturing workspace structure. In the next sections, we describe the CMA and modifications/improvements made for this effort.

4.2.1 Discretization and Randomized Sampling

First, the CMA discretizes the robot's reachable workspace. The workspace is enveloped in a cube centered at the robot arm base with a side length equal to twice the arm length. An overestimation is used to insure that none of the reachable workspace is left out. This cube is then subdivided into smaller cubes which each represent a volume in world coordinates whose reachability must be determined.

The robot's C-space is then randomly sampled using a Monte Carlo method. Each joint value is sampled according to a uniform distribution (i.e. the possible values for each joint are equally likely). Random numbers are generated for each DOF and correspond to a particular joint value. Given the configuration generated, forward kinematics is used to locate the position of the tool center point (TCP). This TCP is then mapped to one of the discrete workspace sub-cubes. The sampling produces an approximation of the reachable workspace after discarding sub-cubes with no valid TCPs.

4.2.2 Generating TCPs and Determining Validity

Once the reachable workspace is determined, dexterity in the workspace is found by generating multiple TCPs at a location in space and then checking their validity. A location with more valid TCPs infers more manipulator dexterity at that location. To generate the trial TCPs, each sub-cube is inscribed by a *reachability sphere* with diameter equal to the cube width. The location of the sphere is determined by its center. Points are equally distributed on the sphere's surface and a TCP frame is generated for each point with the z-axis directed from the point on the sphere's surface to the sphere's center. Furthermore, each one of these frames is rotated about the z-axis by a fixed step-size to generate additional frames. An inverse kinematic solution is computed for each frame. If one of the rotated frames about the z-axis of a particular point on the sphere is valid, only the validity of that point is recorded. Thus, the z-axis orientation of the TCP with respect to the sphere center is disregarded. For a redundant manipulator, the analytical solution is found by optimizing the inverse kinematic solution using appropriate performance criteria. For each reachability sphere, the reachability index D is calculated according to Equation 3-8.

4.2.3 Workspace Directional Structure and Visualization

The z-axes of the valid frames represent a valid grasping approach direction/vector set. If the z-axes are visible on each reachability sphere, the set of axes appear to be grouped in a particular volume of the sphere. Primitives such as cones and cylinders are used to encompass the set of valid axes. Thus, the valid frames are approximately contained within the volumes of these primitives (see Figure 3-7). For example, when using a cone to test if a given TCP on a sphere is valid, the test is reduced to computing the angle between the directional vector and the cone's axis rather than

testing the directional vector against every known and valid TCP on the sphere. This leads to a reduction in computation time and data storage. The CMA defines a shape-fit function that determines how well primitives approximate all valid solutions based on how many reachable points it fails to cover and how many non-reachable points it erroneously covers. A mix of cone and cylinder primitives appears to fit the data the best. [Zacharias, 2007]

One of the goals of this formulation is to also generate a map that visualizes the structure and manipulability in the workspace. The plots that contain both data sets are referred to as capability maps (the visualization tool used in the CMA is not mentioned). *Task planes* are slices of the capability map that do not contain the directional information (see Figure 4-1). Task planes represent a planar subset of the full capability map and show the inner workspace structure which cannot be seen otherwise. While it is possible to visualize the map's entire volume, in most cases task planes are extracted.

4.2.4 Simplifications, Assumptions, and Constraints

There are a number of simplifications that we make to the CMA to develop our approach. One of the biggest modifications is the part of the workspace that is actually mapped. Mapping the entire workspace takes an incredible amount of data storage and computing time. Approximately 12.4 hours were required to generate a finished capability map for the Justin robot. [Zacharias, 2008] Additionally, shape primitives were used to reduce computation time but perhaps more importantly reduce data storage needs.

In order to use the methods operationally, we must dramatically reduce computation time and data storage. This can be accomplished by generating data as needed such as a single task plane or, possibly more generally, a task space. This data

also allows for the visualization of workspace internal structure. Planes of interest could be extracted from precomputed capability maps. However, when applying a map to a task, significant amounts of generated data are not used because of task constraints and much of the work is commonly performed within a smaller workspace volume. Adjacent parallel task planes are unlikely to vary much if they are not located near the workspace boundary. Thus, if we are working in a region of known manipulability, we only need a single plane that cuts through the middle of the task volume instead of all planes that intersect the volume. If more accuracy is needed, generating a few more planes to fill the task volume is not difficult.

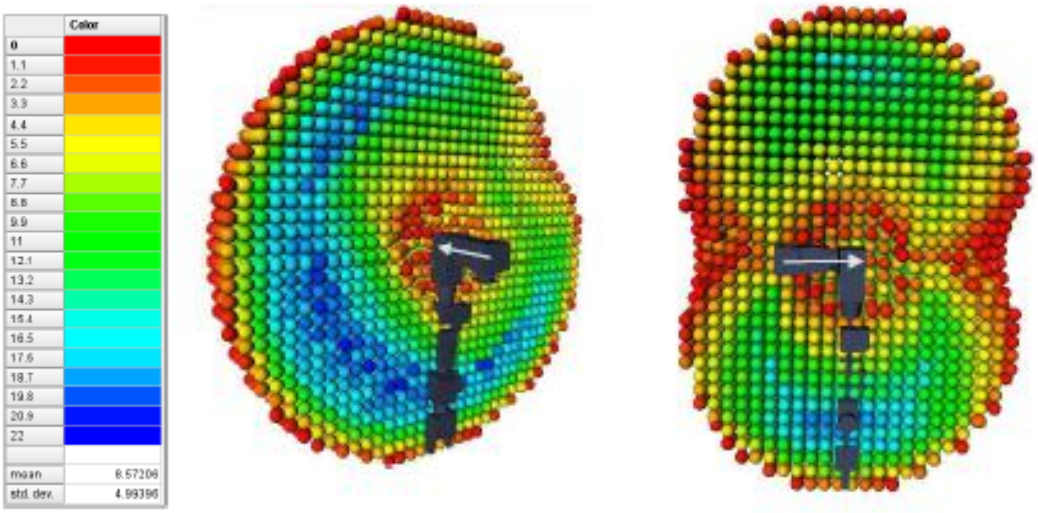


Figure 4-1: Vertical and horizontal slices representing task planes. [Zacharias, 2008] These planes show the internal structure of the capability map.

Manipulator kinematics is of primary concern for our application areas, so this implementation disregards dynamics. We also focus our effort on fixed-base systems with parallel-jaw grippers. The executed grasps will be from the side or from the top, with an additional EEF rotational DOF about an axis normal to the object's supporting

surface. Objects are assumed to be symmetrical and have simple geometry such as cylindrical objects. These assumptions are made to ensure clarity of our demonstrations and to evaluate the approach's effectiveness for many critical applications. The principles of generalization and extensibility will be maintained as part of our implementation to ensure that these algorithms are compatible with more complex grippers and objects of interest.

4.2.5 Extending Capability Map Functionality

The specific application areas motivating this effort (i.e. the nuclear domain, etc) dictate additional constraints the developed algorithm must address.

4.2.5.1 *Cluttered and/or Confined Environments*

Most systems perform multiple tasks. Thus, the objects (i.e. instruments, containers, etc) related to one task may be obstacles when executing another task. The position of these obstacles and targets may also change with respect to time. Additionally, in a more confined environment (i.e. glovebox, lab area, etc), the full volume of the manipulator's workspace may be *severely* restricted. Our proposed method must consider obstacles and other environmental constraints when generating task planes. The necessarily reduced task planes that remain will be a subset of those Zacharias [2007] examined. The areas of reduced manipulability caused by obstacles could have a dramatic effect on subsequent path planning.

4.2.5.2 Nonlinear Trajectory and Grasp Configuration Planning

When performing linear trajectories, Zacharias [2008] used binary maps extracted from task planes. Each binary map corresponded to a single TCP orientation from the task plane, with the location of a reachability sphere in the binary map filled in if the TCP orientation was attainable and left empty if unattainable. In order to find valid trajectory locations in the task plane, a trajectory pattern was searched for in the binary maps. Although binary maps were created for all possible TCP orientations in the task plane, each map was searched independently for trajectories. Therefore, it was assumed that the TCP orientation/EEF orientation was fixed along the entire trajectory.

This fixed orientation constraint excludes many useful motions. Removing this constraint increases problem complexity due to the need to select grasp orientations throughout the trajectory. However, a benefit of its removal is more trajectory options. This also increases flexibility when determining which final grasp will put the manipulator in a more dexterous kinematic configuration. We can incorporate an added layer of planning to determine the best sequence of grasp configurations along the trajectory. This could be accomplished in planar motion if EEF rotation about an axis normal to the plane is unconstrained, such as pick-and-place of a cylindrically symmetric object.

Zacharias [2008] also only planned straight-line trajectories with task planes such as opening a drawer and left generalization to non-linear trajectories, such as opening a door, as a topic of future work. The ability to support non-linear trajectories is very important. This is especially true in more congested environments where non-linear paths must be followed to avoid collisions. Previous RRG efforts to exploit redundancy can be taken into consideration to improve the quality of the task plane and the paths generated.

4.2.5.3 Feasibility for Use in Real-time

We have already discussed the use of task planes for system design and mentioned their potential application in dynamic environments. However, extension to dynamic environments may be complicated by the computational time needed to generate a single plane. How quickly a new task plane needs to be computed will be determined by the frequency at which changes occur and the speed of task execution. If generating a task plane takes too long, then we must devise an alternative way to represent the current manipulator capabilities. This could be accomplished by forming partial planes that only visualize the areas of immediate interest in the workspace or a stored selection of commonly used task planes that can be rapidly read into the control program.

4.3 OUR MAP REPRESENTATION

The Capability Map Approach is the baseline for our approach to generating similar visualization tools for manipulability. As discussed, there are simplifications and some extensions that we incorporate. Thus, our method does not completely follow the CMA, and our adapted method will be referred to as the Modified Task Plane Approach (MTPA). The basic steps of the MTPA are outlined below.

4.3.1 Grid Discretization of Workspace

To generate task planes, our discretization of the reachable workspace uses a 2-D analog to the CMA's 3-D approach. Whereas the CMA had a volume sub-divided into cubes, the MTPA sub-divides an area into squares. This produces a grid defined by the centers of the squares. To aid motion planning, we use a rectangular grid instead of an oval-like one. The size of the grid is based on the reach of the manipulator or the

workspace area of interest. The grid point spacing is determined by the desired resolution. In certain cases, the discretization is further reduced by environmental constraints. For example, generating half of a task plane's grid for a manipulator mounted normal to a wall.

The CMA uses randomized sampling to accurately determine the reachable workspace boundary. This eliminates inverse kinematic computations of discretized points that lie outside the workspace, reducing computation time. The MTPA does not use randomized sampling. If we are only interested in tasks that take place across a few task planes or in a much smaller volume within the reachable workspace, we do not need a precise boundary of the reachable workspace. If we have a rough approximation of the workspace boundary, our overestimated discretization is not as costly because there are fewer computations involving unreachable points. Thus, for the MTPA, randomized sampling of the C-Space is not necessary and allows for computation of the inverse kinematics for all discretized points. Eliminating the Monte Carlo method for randomized sampling also improves the MTPA's real-time operation.

4.3.2 Generating TCPs and Determining Validity

We use the *reachability sphere* concept to generate TCPs/TCP frames at each grid point. We center a reachability sphere at a grid point with an arbitrary diameter less than the grid spacing. In the MTPA, we are currently concerned with grasping directions parallel to and within the task plane. Another grasping direction of interest is when the gripper is parallel to a normal on the plane located at a sphere's center. These two grasp configurations are referred to as *side* and *top grasps*, respectively.

Thus, we do not conduct a search of TCPs that are uniformly distributed on the sphere surface as in the CMA. The sampling method is most similar to the brute force

method used by Zacharias [2006] in Section 3.4.2. The intersection of a reachability sphere and the plane results in a circle with the sphere's diameter (see Figure 4-2). For side grasps, we evaluate grasping direction vectors $\boldsymbol{v}_{gd}$ defined using a point on the circumference of this circle to its center (i.e. the grid point). There is a grasping angle associated with each $\boldsymbol{v}_{gd}$ measured counter-clockwise (CCW) from a reference point on the circle to the intersection of $\boldsymbol{v}_{gd}$ and the circle. This angle specifies $\boldsymbol{v}_{gd}$, and is denoted as the *free angle* or *free Euler angle*, θ_f.

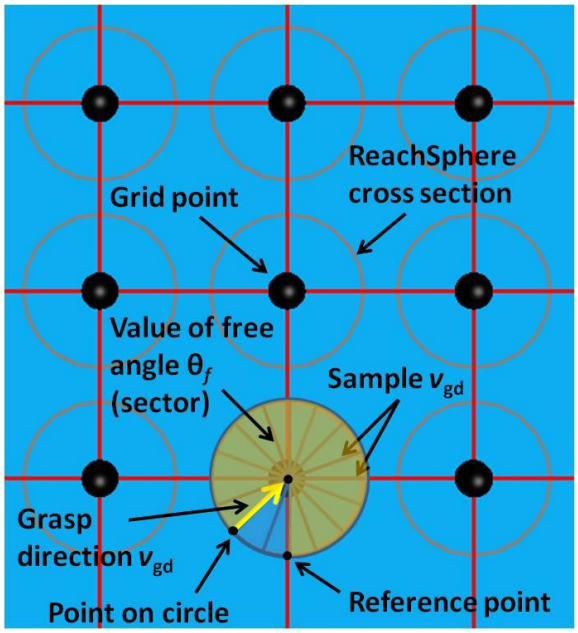


Figure 4-2: Derivation of grasping directions $\boldsymbol{v}_{gd}$. Sample grasp directions $\boldsymbol{v}_{gd}$ are uniformly distributed about the grid point. The $\boldsymbol{v}_{gd}$ being tried is in yellow. The associated grasp free angle θ_f is given by the yellow sector and is approximately 315º in this case.

We then generate a single TCP frame centered on the grid point with a z-axis along the specified $\boldsymbol{v}_{gd}$ and a rotation about $\boldsymbol{v}_{gd}$ that puts the hand in the proper orientation to make a side grasp (see Figure 4-3). (Contrary to the CMA, there are no additional

TCP frames generated by rotating this TCP frame about its z-axis.) We generate TCPs based on $\boldsymbol{v}_{gd}$ uniformly selected around the circle's circumference creating a set of trial TCPs for the hand. For top grasps, we again center the TCP on the grid point, with $\boldsymbol{v}_{gd}$ used as a reference vector to rotate the TCP by a fixed amount about the normal vector. Each $\boldsymbol{v}_{gd}$ corresponds to a particular TCP orientation about the normal, generating another set of trial TCPs. For this effort, the rotation can vary from 0º to 180º since the grasp configurations are symmetrical.

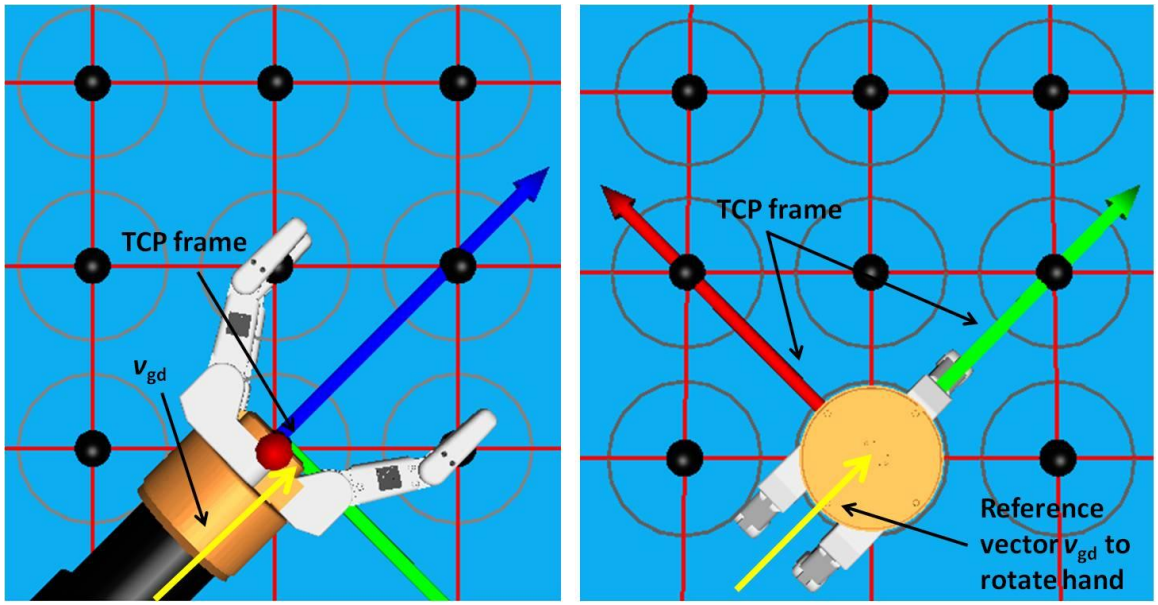


Figure 4-3: Demonstration of side and top grasps made using the same yellow $\boldsymbol{v}_{gd}$ as in Figure 4-2. For the side grasp, the TCP frame is centered on the grid point and $\boldsymbol{v}_{gd}$ aligns with the z-direction of the TCP frame. For the top grasp, the z-axis of the TCP frame is into the plane and normal to it. Here, $\boldsymbol{v}_{gd}$ aligns with the TCP frame y-axis.

The validity of each $\boldsymbol{v}_{gd}$/TCP is determined using inverse kinematics. Then we perform a joint limit and collision check on that joint configuration. A final check determines if a joint-interpolated motion plan can be made from a predetermined manipulator configuration to each $\boldsymbol{v}_{gd}$/TCP. The same reference configuration is used for

all $\boldsymbol{v}_{\mathbf{gd}}$/TCPs in a task plane. By confirming the existence of a valid motion plan, we confirm that the configuration is both *physically **and** operationally* attainable. If there are physically attainable positions that are not operationally attainable, one could argue there are deficiencies in the algorithm calculating the inverse kinematics (i.e. algorithmic singularities, uncontrolled joint drift, etc.), but such deficiencies that do exist must be addressed. (The notion of using the ratio of physically and operationally attainable points to measure the quality of an inverse kinematics algorithm is noted and left for future work.) If for any TCP, the corresponding manipulator configuration collides with its environment, exceeds joints limits, or the motion plan fails, then that TCP is not valid. For example, a grid point that overlaps with any part of an obstacle will have no valid TCPs. The *manipulability index M* is then calculated using the same formula as the reachability index (Equation 3-8),

$$M = \frac{R}{N} \cdot 100, \tag{4-1}$$

where R is the number of valid TCPs and N is the total number of points on the circle's circumference at a grid point. Although the mathematical formula is the same for both indexes, we use the term *manipulability index* to distinguish between the methods that produce each index.

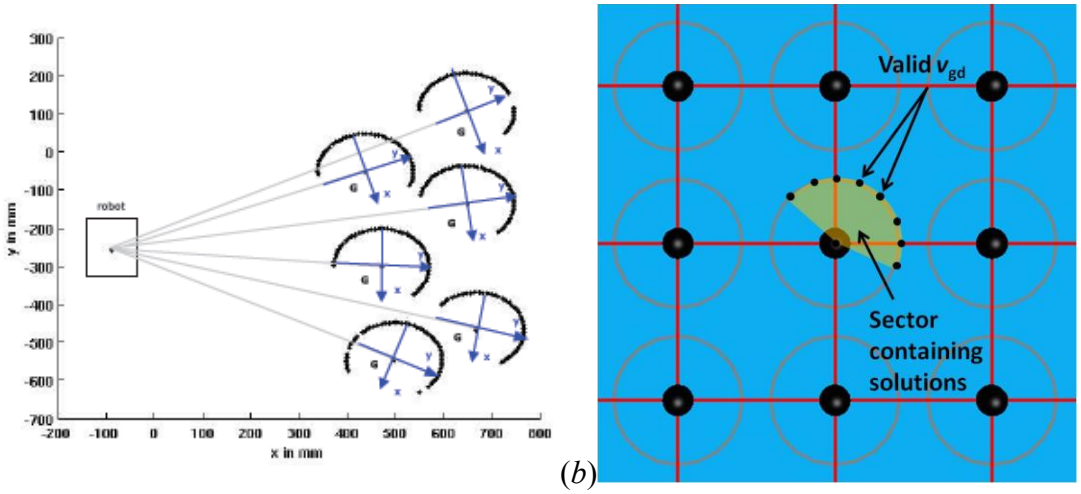
4.3.3 Workspace Directional Structure and Visualization

With our 2-D simplifications, 3-D primitives to capture preferred $\boldsymbol{v}_{\mathbf{gd}}$ will not be necessary. In the plane, groups of adjacent valid TCPs will be contained in sectors – the 2-D analog of a cone. However, we do not need shape-fitting of valid TCPs to determine

the range that valid angles belong in. Establishing the largest continuum of valid inverse solutions is similar to the technique discussed in Zacharias [2006] (see Section 3.4.2). Information about this continuum will be utilized in our approach to motion planning.

For clarity, visualization of the results is split into plots containing manipulability and directional information respectively. The task plane plot contains the manipulability indices from all grid points and does not use the reachability sphere shape in the visualization. This plot can take a 3-D or 2-D form. A 3-D plot contains x and y values that give the locations of grid points in the plane and z values given by the manipulability index at each grid point. Point color based on z values can also be added, with values interpolated between grid points. This generates a potential field plot where higher "potentials" reveal areas of higher manipulability and valleys show areas where manipulability is limited based on dexterity (structure similar to Figure 3-14). We will see later that a color-based 2-D plot has more utility for application purposes. In these plots, the x and y values give the location of a grid point and the value of the manipulability index determines the color of a point/region around a point.

Directional information can also be plotted for side grasps. Sets of TCPs were determined from $\boldsymbol{v}_{\mathbf{gd}}$ emanating from uniformly distributed points on a circle to a grid point. For valid $\boldsymbol{v}_{\mathbf{gd}}$/TCPs, we plot the point on the circle used. We reduce the circle radius as necessary to avoid clutter between adjacent grid points. The visual is similar to Figure 3-20, but with valid points filled in instead of left blank (see Figure 4-4). Valid points are contained by arcs on the circle, leading to a sector of the circle where valid $\boldsymbol{v}_{\mathbf{gd}}$ and their corresponding θ_f are found.



(*a*) (*b*)

Figure 4-4: Two ways of representing valid grasping directions. Image (*a*) is from Zacharias [2006]. Image (*b*) is a visual showing the valid $\boldsymbol{v}_{gd}$ represented as points on the circumference of the circle that $\boldsymbol{v}_{gd}$ emerges from. A yellow sector contains all the valid θ_f and ranges from about 67.5° to 225°.

The grasping direction is the same for all sets of top grasps (i.e. normal to the plane and from above), so a directional plot based on this information is not useful. The visual could instead represent the valid rotations of $\boldsymbol{v}_{gd}$ about the normal vector. For redundant manipulators with roll joints as the final joint, if one top grasp is attainable, then the other grasp orientations are usually attainable by rotation of the last joint. Thus, the typical "all or nothing" validity of top grasp configurations limits the usefulness of this representation and it is not used for these grasps.

4.4 SUPPLEMENTAL CAPABILITY

Section 4.3 outlines our Modified Task Plane Approach and these changes (and their impact) are summarized in Table 4-1:

Table 4-1: MTPA modifications

Modification description	Impact to computational time
Generating task planes/areas not workspace volumes	Significantly reduced
Trial TCPs for side and top grasps only	Reduced by a factor of ~10
Incorporating obstacles/collision detection for TCPs	Nominally increased
Performing joint limit and motion checks for TCPs	Nominally increased
Capturing valid $\boldsymbol{v}_{gd}$ in sectors not 3-D primitives	Reduced by a factor of ~10

In order to use the MTPA for our applications, other capabilities must be developed. These capabilities are related to workspace configuration optimization and motion planning.

4.4.1 Target Modeling

For system design, we need to represent objects to be grasped and volumes of interest where tasks occur. These representations can be reduced to simple geometric shapes such as a sphere, cylinder, or box. Since we are dealing with planes, the intersection of these objects with a plane produces circles or rectangles. Thus, 3-D task volumes and objects are approximated by 2-D circles and/or rectangles. We will use the identifier "target" when referring to task volumes or objects in the plane.

4.4.2 Criteria Calculations

We also need to develop a criterion to compare task planes and target locations in planes. Given a target or task plane, we calculate the *average manipulability index* (AMI) for all grid points contained within a representative shape (see Figure 4-5),

$$\gamma_{AMI} = \frac{1}{N_g}\sum_i M_i, \tag{4-2}$$

where M_i is the manipulability index of grid point i and N_g is the total number of grid points in the area of interest. Essentially, a larger AMI denotes greater manipulability for a target located in that area of the grid. The AMI over the entire task plane or a task area can also be used to determine which task plane/area is best in general for tasks that are yet to be defined.

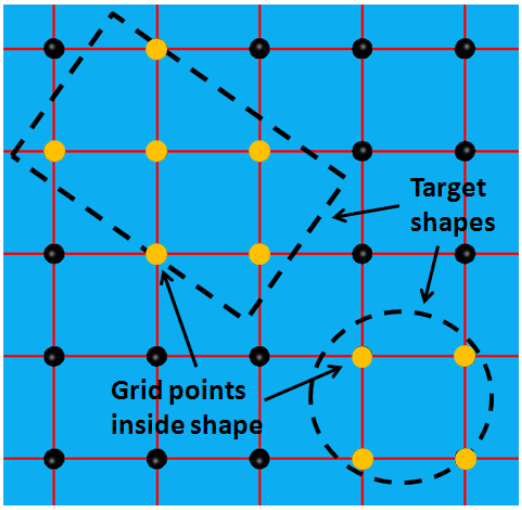


Figure 4-5: Two target shapes with the grid points that lie inside them in yellow. The AMI for each shape is found by summing the manipulability indexes of all the spheres in yellow and then dividing by the total number of yellow spheres.

With our target and criterion formulation, we can also perform the same analysis that motivated Abdel-Malek's efforts to identify manipulator base locations that allow for the most dexterous manipulation of targets. [2004] In the MTPA, we assume a fixed manipulator base and find the best target locations. These two problems are essentially identical as both find the best relative positions of the targets and base for manipulation.

To perform our search, we generate a task plane then overlay the targets at various locations in the grid, calculating the AMI for each iteration. After trying all possible configurations, we know the best location of the targets within a task plane in regard to manipulability. Repeating for multiple parallel task planes, can give an even better idea of the optimal placement of the targets in the workspace.

4.4.3 Trajectory Functionality

As previously mentioned, Zacharias [2008] applied capability maps to motion planning. In this case, motion was limited to a pre-defined linear trajectory in a task plane where the same TCP orientation was maintained during the entire trajectory. From the chosen task plane, multiple binary maps were constructed based on different TCP orientations. Thus, a single binary map only contained reachability spheres where a particular TCP was valid. Possible trajectory locations in the binary map were found by using a pattern search algorithm that looked for various orientations of the trajectory. The search resulted in linear paths that maintained the same TCP orientation throughout.

The MTPA can also be applied to motion planning, but does not use binary maps and pattern recognition techniques. The MTPA lends itself to the construction of a connected graph and an approach similar to the skeleton method presented in Section 3.3.4. The grid points can represent nodes in a graph, where a grid point is connected to adjacent points directly above and below and to the left and right. A point is not connected to an adjacent point if the adjacent point's manipulability index is zero. We then choose a starting and ending grid point and use a search algorithm to find a connected path between those two nodes. If the initial and/or final position is not located on a grid point, we can plan a motion to the nearest grid point and then perform the search.

Due to the nature of our graph, several collision-free paths are likely to exist. The path selected is based on the manipulability index. A shortest path algorithm creates a cost of $100 - M_i$ for moving to grid point i. Thus, a point with a lower cost has a higher manipulability index. Although our goal is to identify a valid path, the skeleton approach provides the added benefit of finding the best path of multiple paths based on a measure of manipulability (i.e. the path found has the highest AMI). The AMI can also be used as a tool to determine the quality of generated paths. Zacharias' approach could find the region of the workspace where a valid trajectory was found the most (Figure 3-10), but there was no indication of which trajectory was best.

Due to the nature of our tasks, we use this graph method to perform Cartesian space planning. Again, using the MTPA differs from the CMA of Zacharias [2006] in two important ways: we are not constrained to linear trajectories and any valid trial TCP frame is allowable while traversing the path. The search algorithm finds the path to follow, and a motion planner plans the hand motions between nodes. The search algorithm does not identify TCP frames for each node, but we can use information about the valid $\boldsymbol{v}_{gd}$ at each grid point to make this decision. Essentially, this information indicates good configurations at a point in the workspace regardless of whether the robot is making a grasp or moving through space. Valid $\boldsymbol{v}_{gd}$ lie in one or more sectors about a grid point and the largest sector contains the longest continuum of valid $\boldsymbol{v}_{gd}$. Choosing the $\boldsymbol{v}_{gd}$ in the middle of this sequence to form the TCP frame at that grid point should give a fairly stable manipulator configuration. The TCP is selected in this manner for all grid points in the path, and the planner determines the poses between adjacent TCPs to form the full path from start to finish.

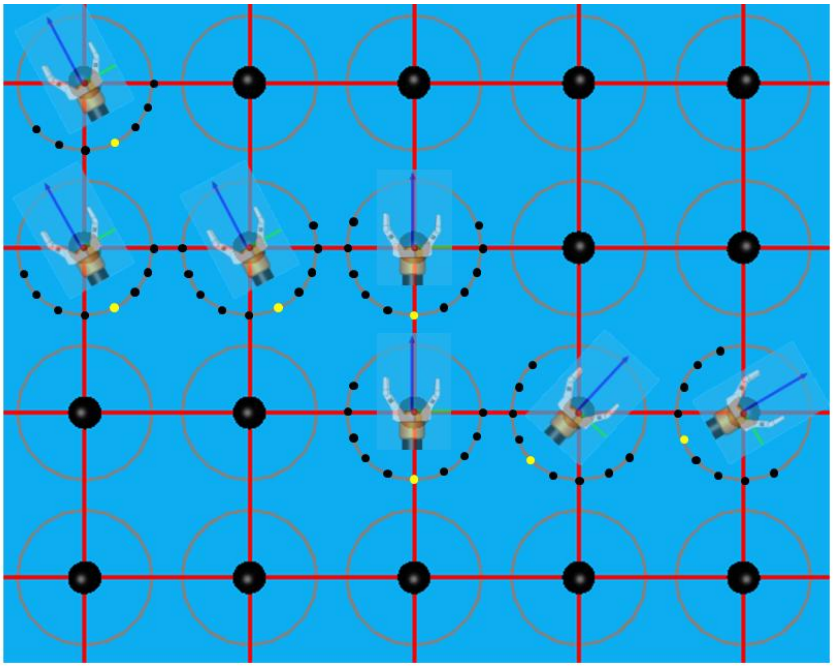


Figure 4-6: Demonstration of selected TCPs for motion planning. A path is found based on manipulability from the top left corner to the bottom right. The yellow points on the circles show the starting point of the $\boldsymbol{v}_{gd}$ in the middle of the longest continuous string of valid $\boldsymbol{v}_{gd}$. These $\boldsymbol{v}_{gd}$ are superimposed by the TCP orientation the hand takes to achieve the grasp.

Other strategies could be employed to determine a valid path. One of the objectives of the implementation presented in Chapter 5 will be easily modifying or extending the MTPA to include such strategies. Lessons learned during demonstration and testing may lead to other strategies discussed in Chapter 6.

4.5 APPROACH SUMMARY

We have covered the details and applications of the Modified Task Plane Approach (MTPA). Below is a simplified flowchart of the method:

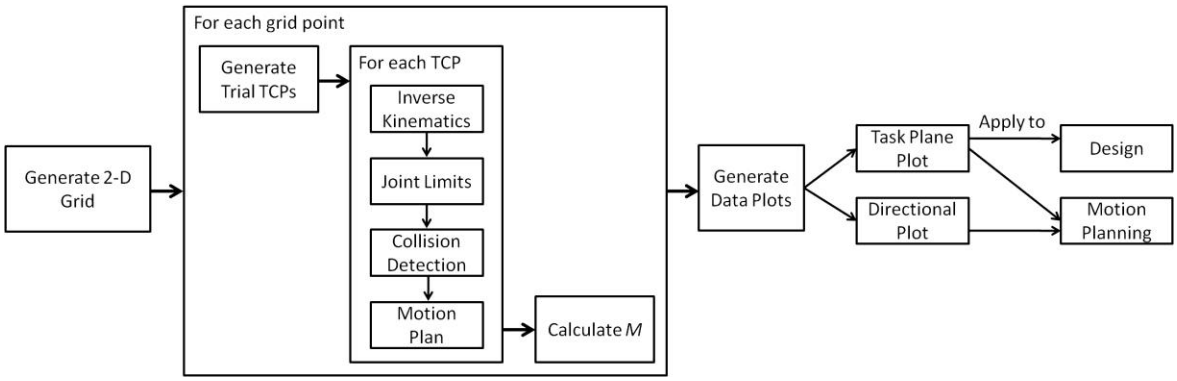


Figure 4-7: MTPA flowchart.

In the next chapter, we will discuss our implementation of this method in order to perform a number of design and motion planning tests.

Chapter 5 **Implementation and Demonstration**

5.1 HIGH LEVEL IMPLEMENTATION

At a high level, we are creating a map of a manipulator's workspace that represents its manipulation capabilities. This is done by uniformly sampling the workspace and determining the valid grasping configurations at each sampled location. This produces a quantitative measure of the dexterous workspace that is used as an evaluation tool for system design and in our grasping strategy for task completion in dynamic environments.

5.2 CODE DEVELOPMENT USING OSCAR

The lower level implementation will use Operational Software Components for Advanced Robotics (OSCAR) which was developed at The University of Texas. [Kapoor and Tesar, 1996] OSCAR is a C++ framework for the development of control programs for robotic manipulators and provides many operational capabilities:

- Foundational mathematical and data support
- Forward kinematics
- Inverse kinematics (with support for redundant manipulators)
- Motion planning (with support for collision detection and obstacle avoidance)

In OSCAR, manipulator data (D-H parameters, joints limits, etc) and environment data (obstacle and manipulator models) is stored as a workcell using an XML file (see

Figure 5-1). OSCAR workcells are used with this research and other OSCAR functionalities utilized by this effort are discussed when applicable.

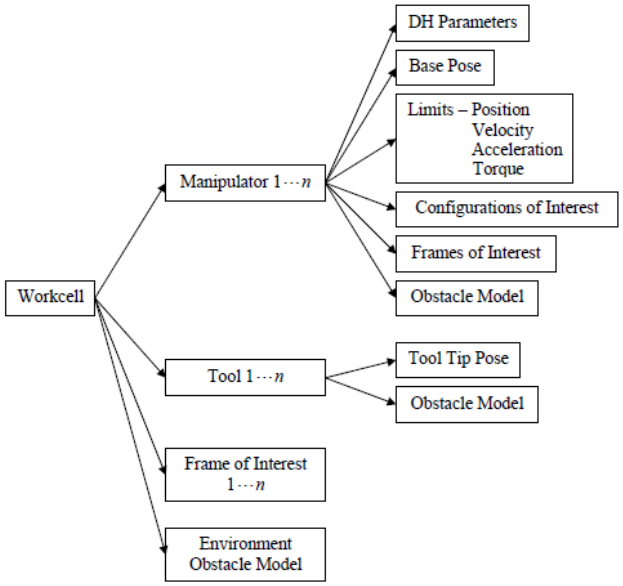


Figure 5-1: Structure of the workcell in OSCAR with information about the manipulator and its environment. [Knoll and Tesar, 2007]

Many methods for quantifying manipulability have already been implemented and used in operation. [Lee, 2006][Pholsiri, 2004] OSCAR also has a module for performance criteria that uses some of the metrics detailed in Chapter 3 regarding manipulability/dexterity. These performance criteria include the Measure of Manipulability (based on the determinant of the Jacobian), Velocity Transmissibility, Force Transmissibility, Joint Range Availability, and Singularity Avoidance (based on the minimum eigenvalue of the Jacobian). Capabilities exist in OSCAR to fuse criteria and rank joint solutions for redundant manipulators. These criteria can also be utilized to compare orientations at a location in space where there are fewer task constraints than available DOFs or to rank possible configurations perturbed about a current

configuration. When using criteria, the result is often given as a number that designates the quality of a selection, but how this relates physically to the system is often lost. The CMA and MTPA visualize the type of results usually only given numerically. Although the MTPA incorporates the AMI criterion, the relation of this criterion to the physical system is straightforward.

5.2.1 OSCAR Extensions

As a part of this effort, OSCAR's functionality must be extended to include new capabilities that support task planes. The `ReachSphere` and `Target` classes constructed will be discussed shortly. In regard to motion planning, OSCAR contains no libraries related to graph construction and traversal. Thus, this functionality is obtained from an outside source. We could have incorporated a holistic outside source, such as the BOOST graph libraries, but we did not need that level of capability. For our purposes, we needed graph construction that supported the *A search algorithm, and we found prewritten code [Heyes-Jones, 2001-2005] that could be modified to interface with OSCAR.

5.2.2 Task Plane Implementation

The MTPA outlined in Section 4.3 was implemented in C++ using OSCAR. The size and boundaries of the workspace grid depend on the location of the base frame and the desired workspace region to analyze. The task planes are defined as x-y grids with the positive z-axis pointing straight up. The base frame and grid point locations are defined relative to the global frame. Grid points are equally spaced with a common spacing between adjacent points of 2.5 cm or 5 cm. The closer the points are, the more

likely that there are only small changes in valid grasps between adjacent points. However, in some parts of the workspace, there can be a steep drop off in manipulability that may render this assumption invalid. Going below 2.5 cm, the trade-off between better resolution and computational time is not very beneficial. A spacing of 5.0 cm was set as the upper limit. From initial trials, if the spacing is too large, narrow target representations can actually fit between the grid points and thus do not register any overlap. The grid spacing is adjustable if more or less resolution is needed for a particular task plane application.

5.2.2.1 Reachability Sphere

Once the grid is established, we need to generate and determine the validity of TCPs at each grid point. A `ReachSphere` class was created in OSCAR to contain all the information about manipulability at a point. This object has the following members:

`ReachSphere`

- Unique sphere ID (numerical tag)
- Global x, y, z position
- Number of free Euler angles θ_f to sample
- Array of free Euler angles θ_f that make valid TCPs
- Manipulability index M

When creating the `ReachSphere` object, the input parameters are the sphere ID and global position. The class has supporting functions for filling the TCP array and determining the manipulability index of a grid point.

The main `ReachSphere` function used in task plane generation is `BuildSphere`. This function generates the TCPs to try at a grid point and performs the inverse kinematic computations to check for reachability/validity. This function also performs the joint limit, collision, and motion plan checks. In OSCAR, TCPs are stored as `HandPose` objects that contain the x, y, and z locations of the tool point and three Euler angles that define its orientation. When forming a `HandPose` to test, the position is set to the global x, y, and z position of the `ReachSphere`. Depending on what type of grasp is attempted (side or top), one of the three Euler angles defining the hand orientation is free to change (θ_f). The other two Euler angles set the `HandPose` so that the gripper is in the correct orientation to make the grasp. The desired EEF position and orientation given by the trial `HandPose` will be defined as $\boldsymbol{u}_{des}$.

To develop $\boldsymbol{u}_{des}$, the free Euler angle ranges from 0° to 355º, with the positive direction being CCW about the positive z-axis starting from the negative x-axis. The angles are assigned in 5° increments yielding 72 generated $\boldsymbol{u}_{des}$ values. Thus, each angle has an index from 0 to 71 in the θ_f array contained in the `ReachSphere` object. Depending on the needed resolution, the angle increment, and thus the number of $\boldsymbol{u}_{des}$, can be adjusted. An OSCAR `IKJReconfig` or `IKJAvoidLimits` inverse kinematic object (IK) is commonly used for redundant manipulators. The latter IK uses the Joint Range Availability criterion in optimizing a manipulator's kinematic configuration.

Next, joint solutions $\boldsymbol{\varphi}_i$ are found using inverse kinematics for each $\boldsymbol{u}_{des}$. If a $\boldsymbol{u}_{des}$ is valid, the θ_f array in the `ReachSphere` object is populated with the value of θ_f. Otherwise, a value of -1 is assigned, denoting an invalid $\boldsymbol{u}_{des}$. There are joint limit and collision checks for valid configurations. We eliminate configurations that cause any part of the robot to collide with an obstacle. Motion plans are generated using a `FifthOrderPoly` object from a reference `HandPose` $\boldsymbol{u}_{ref}$ to the destination pose

using a 5th order polynomial. If the $\boldsymbol{u}_{des}$ fails one or more of these checks, it is invalid. The `BuildSphere` function also has the functionality to output the $\boldsymbol{\varphi}_i$ that are found from inverse kinematics for each valid $\boldsymbol{u}_{des}$.

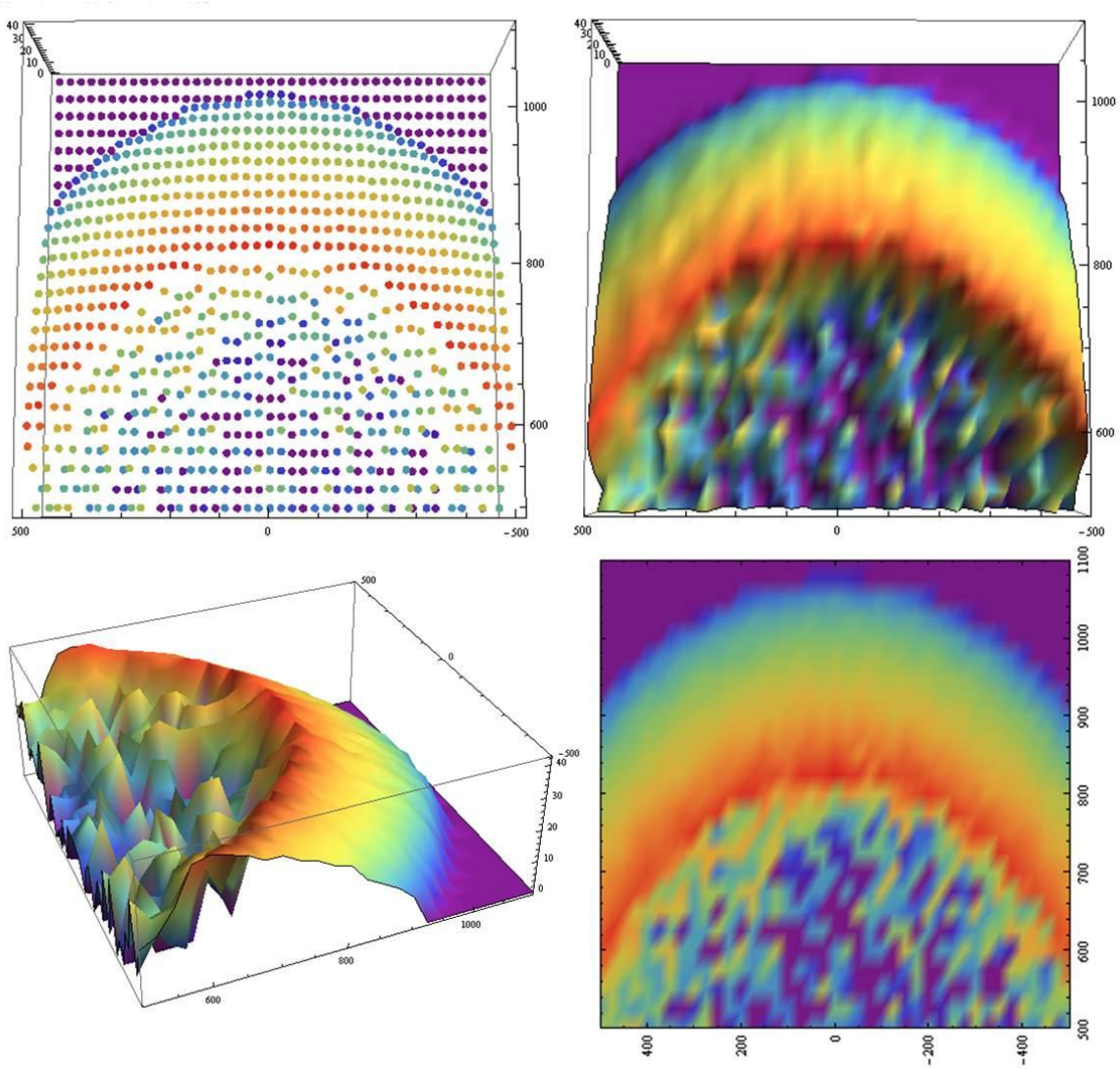


Figure 5-2: Example representations for the same task plane. Three-dimensional plots are shown in point and surface form. The value of x ranges from 500 mm to 1100 mm, and y ranges from -500 mm to 500 mm. The height is determined by *M*, which has a maximum value of about 40%. The task plane height is 600 mm with a grid spacing of 25 mm. The coloring is based on a scale normalized to the maximum z value. A 2-D plot (bottom right) is used for application purposes.

5.2.2.2 Targets

The `Target` class is another new OSCAR class. A `Target` object contains the following information,

`Target`

`enum TShape: RECT` (rectangle)

- Global center location: x, y, z
- Length and width
- Rectangle corners p_1, p_2, p_3, p_4 (each an OSCAR::Vector3)
- Rotation about center (relative to global axes)

`enum TShape: CIRCLE`

- Global center location: x, y
- Radius

When a `Target` is substantiated, its center is positioned relative to a local frame initially aligned with the global frame. For a rectangle, the corners p_1, p_2, p_3, and p_4 are then populated according to the length and width arguments of the constructor and rotated about the center as defined by the rotation argument. If only one `Target` is defined, then the shape is centered on the local/global frame. When a group of `Targets` is defined, the centers of these shapes are oriented relative to one another about the local/global frame. The local frame is used as a handle for moving `Targets` throughout the grid in a search that will be discussed shortly.

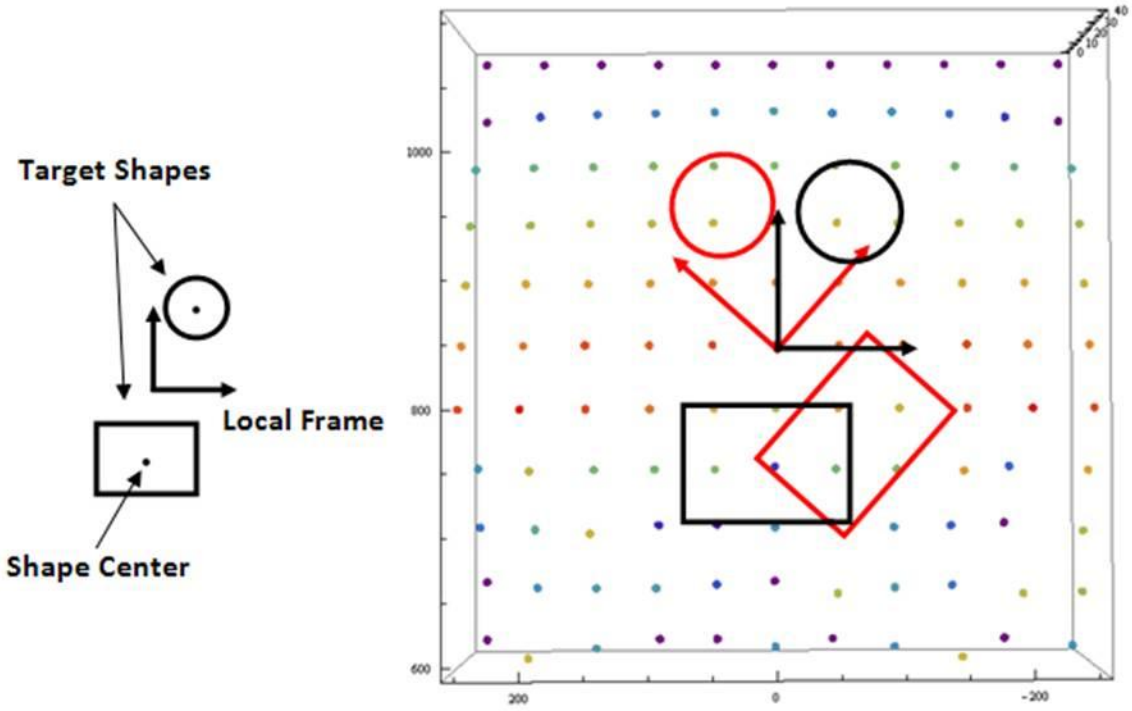


Figure 5-3: Task plane with `Target` shapes. Left: A group of `Targets` oriented about a local frame. Right: The local axis of the `Targets` is centered over a grid point to calculate the AMI. A rotation of the `Targets` about the same grid point is shown in red. The centers of the `Targets` are removed for clarity.

`Target` shapes are used to determine the best target location relative to the manipulator base. The `MoveToGridPoint` function copies the `Targets` then centers their local frame at the desired grid point such that the x-axes of the local and global origins are parallel. The value of the `Target` centers and/or corner points in the global frame is then determined. Since `Target` locations relative to the local frame are known and the local frame is now located on the grid point, the position of the `Targets` relative to the grid point (and the global frame) is straightforward. Then we calculate the AMI of the grid points inside or on the boundary of the `Target` shapes. At each grid point, we also rotate the local frame about its z-axis and calculate the AMI for each

rotation to determine the best orientation of the `Targets` about that grid point. We perform these iterations at all grid points to determine, based on manipulability, the best grid point location and orientation for the `Target` local frame. To reduce the computational time, at each iteration we find $x_{\min}$, $x_{\max}$, $y_{\min}$, and $y_{\max}$ for the `Targets` and only check if a grid point is inside if it is within the area defined by these limits. We also do not check a `Target` orientation at a grid point if any part of a `Target` is outside the grid boundaries. For the following discussions, let us assume that a `Target` object has been populated for each *target*.

Determining whether a grid point is within a circle is straightforward using the equation for a circle. Given the circle's center (x, y) and the grid point's center (x_0, y_0), the point is inside the circle if,

$$(x-x_0)^2+(y-y_0)^2 \le R^2, \tag{5-1}$$

where R is the radius of the circle. The check for a rectangle is more complicated because the rectangle sides could be skewed relative to the global axes (it is not rotationally symmetric). The rectangle `Target` is represented by four corners in the plane, p_1, p_2, p_3, and p_4, to assist with this computation. Starting at p_1, we make a vector $\boldsymbol{v}_{12}$ from p_1 to the next corner clockwise (CW) from p_1 which is p_2 (see Figure 5-4). A second vector $\boldsymbol{v}_1$ is made from p_1 to the center of the grid point of interest. If we generate such a vector pair for each of the four corners and take the cross product of each pair,

$$\begin{aligned} \boldsymbol{v}_{12} \times \boldsymbol{v}_1 &= z_1 \hat{z}, \\ \boldsymbol{v}_{23} \times \boldsymbol{v}_2 &= z_2 \hat{z}, \\ \boldsymbol{v}_{34} \times \boldsymbol{v}_3 &= z_3 \hat{z}, \\ \boldsymbol{v}_{41} \times \boldsymbol{v}_4 &= z_4 \hat{z}, \end{aligned} \tag{5-2}$$

we obtain four vectors in the z-direction. If the sign of all four vectors z_1, z_2, z_3, and z_4 is the same, then the grid point is within the rectangle – mixed signs indicate it is outside.

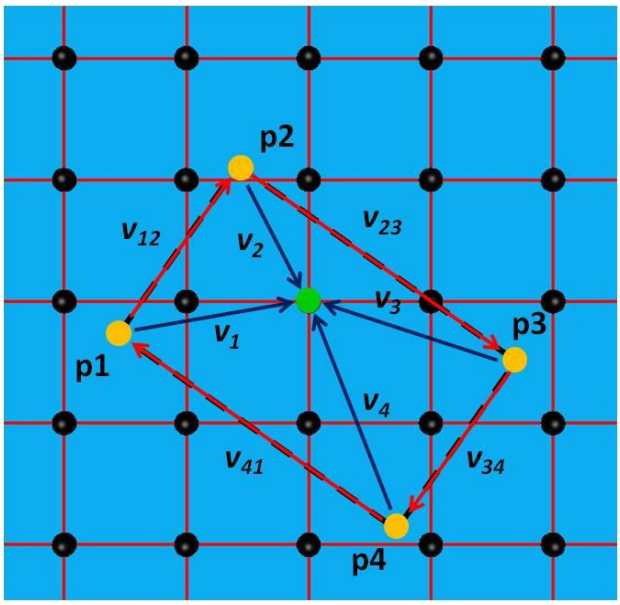


Figure 5-4: Vector formulation for the cross product method that determines whether a grid point is within a `Target` shape. The green grid point is the point of interest.

5.2.3 Motion Planning Implementation

We can demonstrate that task planes are a useful tool for motion planning. Particularly, the manipulability index can help identify paths that lead the arm through workspace regions of greater manipulability. Additionally, the path plan generates good kinematic configurations throughout the entire motion.

5.2.3.1 Graph Construction and Search Algorithm

The MTPA lends itself to motion planning using a graph technique. Building this type of capability into OSCAR would be time consuming and is not a goal of this research. We found an *A algorithm coded in C++ that uses the standard template libraries. This code was developed by Justin Heyes-Jones [2001-2005]. The code is distributed throughout three files: `fsa.h`, `stlastar.h`, and `findpath.cpp`. The `fsa.h` file handles memory allocation and management, `stlastar.h` contains the classes for graph support and searching, and `findpath.cpp` contains the class with functions for feeding map information defined in the control program to graph nodes used by the *A algorithm. This file also contains the code for performing the search in a `main` function.

This code was chosen based primarily on its use of the *A search algorithm to search a grid-like map of cells. Each cell contains a value that represents the cost of moving to that cell. Visually, this code searches for the shortest cost path of a 2-D grid of cells where each cell is connected to the cells directly above and below and to the left and right, unless denoted otherwise (such as a border cell or an unattainable cell). Mathematically, this 2-D grid is stored in a 1-D array which lists data row by row.

Some modifications were necessary for operational compatibility with our control program. The cell/grid map was originally given as a global variable in `findpath.cpp`. We removed the global status of the map so it could be written by the control program. By doing this, we had to make the map an argument to some of the functions used by the algorithm in `findpath.cpp`. Additionally, the cell costs ranged from 0 to 9 (unattainable) and some functions had this range hard-coded into them. We rewrote these functions to handle values from 0 to 100 to support the range of *M*. Essentially, instead of using cells, we used grid points, and instead of using the cell cost

to populate the map, we used 100 – *M*, as previously mentioned. The `findpath.cpp` file also contained a `main` function for performing the search. This code was transferred to our control program, and the resulting code in `findpath.cpp` was renamed as `PathFinder.h`.

When the A* algorithm determines the nodes in a path, the (0, 0) node begins at (x_{max}, y_{max}) in the task plane. Thus, as x and y are decreased, the values of the node positions are increased. This convention was used so that when the task plane is oriented with a vertical x-axis and horizontal y-axis, the first node position value corresponds to x and the column number, while the second node position value corresponds to y and the row number.

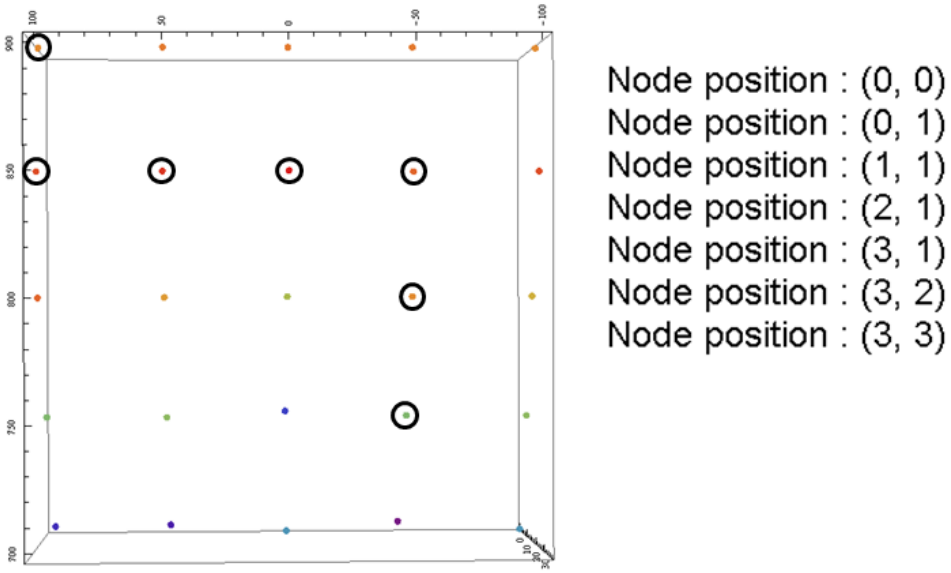


Figure 5-5: A path found in a small section of a task plane. The x-direction is vertical and ranges from 700 mm to 900 mm (bottom to top), and the y-direction is horizontal and ranges from -200 mm to 200 mm (right to left). For the path solution on the right, the x-value gives the column and the y-value gives the row (both starting from the top left corner). The points look slightly misaligned because the view is looking down on a 3-D plot.

5.2.3.2 Cartesian Path Planning

Once the A* map is populated with M values from the grid, the map search gives a sequence of grid points to follow given starting and ending points in the task plane. We utilize the technique described in Section 4.4.3 to determine which θ_f is used to generate $\boldsymbol{u}_{des}$ at each grid point. This task is accomplished by calling the `FindSector` function on each `ReachSphere` object. After the $\boldsymbol{u}_{des}$ are generated, the OSCAR motion planner generates a trajectory that moves through these $\boldsymbol{u}_{des}$. However, the motion planner must plan the hand poses between grid points, and it is not guaranteed that all the joint configurations from these hand poses will be valid. The actual data fed to the motion planner is a vector of OSCAR `Xform` objects. These `Xform` objects are 4 x 4 transformation matrices that transform the tool frame to $\boldsymbol{u}_{des}$. They can be determined using $\boldsymbol{\varphi}_i$ and forward kinematics.

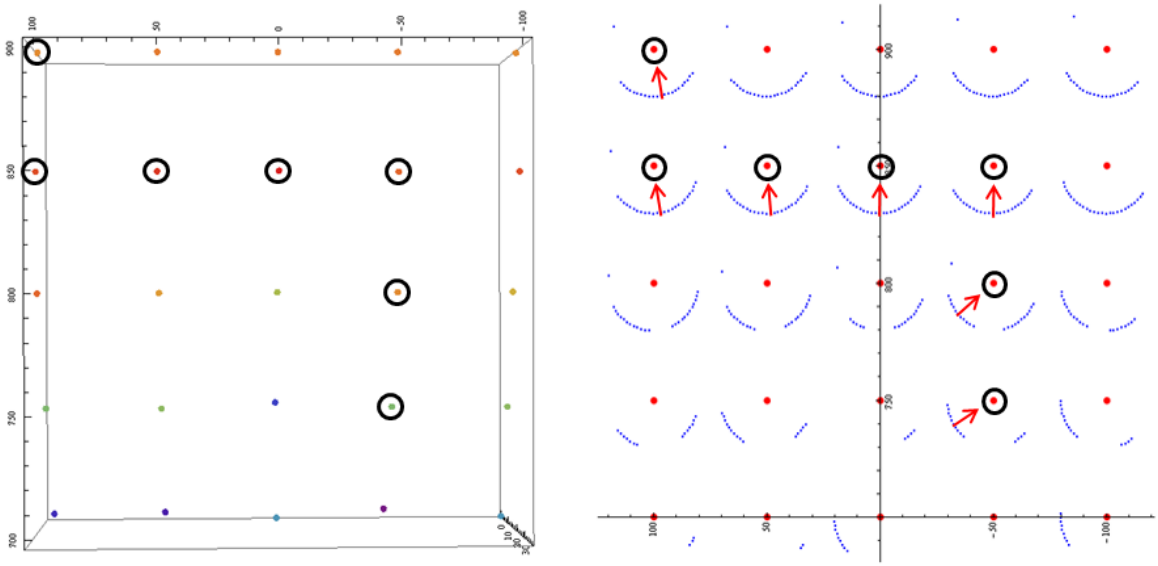


Figure 5-6: The path found in Figure 5-5 and the corresponding directional plot using the same grid points. The red arrows in the directional plot show the $\boldsymbol{v}_{gd}$ selected at each grid point to develop the sequence of TCPs/Handposes sent to the motion planner.

The resulting trajectory appears to be fully constrained as it moves between $\boldsymbol{u}_{des}$ at the grid points. However, the motion is unconstrained between $\boldsymbol{u}_{des}$, and large displacements in $\boldsymbol{\varphi}_i$ and various hand orientations are possible. The small motions between constrained $\boldsymbol{u}_{des}$ limits these types of movements, but they could still occur and should be checked. Also, the OSCAR motion planner does not check for joint limits in the trajectory it generates, so although a path and trajectory are found, the motion may not be physically achievable.

5.2.4 Implementation Issues

There were a few issues with OSCAR that limited the implementation of our approach. Our version of OSCAR lacked the capability to use the SOLID libraries for collision detection. Thus, models were made using primitives. When checking the validity of TCPs, we wanted to incorporate a more thorough motion planning check, one that used the OSCAR motion planner. However, there was a bug in the OSCAR application code that produced errors when the motion planner was run in the iteration loop that checked the validity of all trial TCPs. Additionally, for each valid TCP, we could have stored the $\boldsymbol{\varphi}_i$ found by the IK for later use with the motion planner. This required an immense amount of data storage and frequent access errors resulted, so this was not implemented.

5.3 EXPERIMENTS

Once we built a program with the capabilities discussed in Sections 5.2.2 and 5.2.3, we ran a number of experiments. Unless otherwise mentioned, the base frame is located on the physical bottom of the manipulator, coincident with the first rotational

joint (Figure 5-7). The end-effector is a 3 fingered Barrett gripper used in a simple parallel jaw configuration that can grasp from the side or from above. The tool point (TCP) is defined along the z-axis of the last joint frame and was chosen based on the hand location. The hand was positioned a reasonable distance from the last joint of each manipulator given the dimensions of the robot. The tool point is positioned 50 mm from the center of the hand. Task plane grids are constructed in the x-y plane at a given height in z. Manipulator and obstacle modeling is performed using OSCAR `Primitives` such as `Cylispheres` and `Planes`. In many cases, this modeling exaggerates the actual physical dimensions of the manipulator, and is thus a conservative approach.

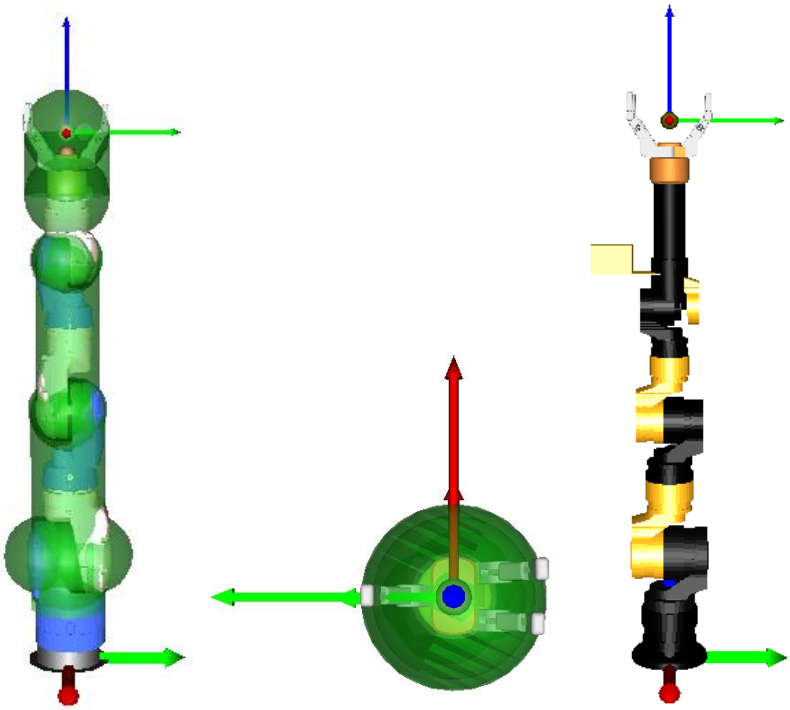


Figure 5-7: Manipulators showing the base frame and tool point for zero joint displacements. The Motoman SIA-10 arm (left) and the PowerCube LWA3 arm (right) are shown. A zoom-in of the hand shows the position of the tool frame relative to the fingers of the hand (center).

5.3.1 Application to Design

One of the main objectives is to apply the MTPA to system design problems. We want to demonstrate that task planes are effective in making decisions about various design options. These include the layout of the manipulator and targets in the workspace and manipulator selection. Task planes have already been used to position mobile manipulators for tasks (see Figure 3-9), [Zacharias, 2007] and we can show this same ability and others with the MTPA.

5.3.1.1 Configuration Management of Glovebox Manipulation

We use task planes to determine a quality manipulator base location to manipulate targets in a glovebox environment. The glovebox used for these experiments is shown in Figure 5-8. The modeled glovebox is similar to some used at LANL and is 1625.6 mm long by 1371.6 mm wide at the base and 1143 mm tall. The global frame is in the bottom left corner of the box, and the manipulator is a Motoman SIA-10 7 DOF arm.

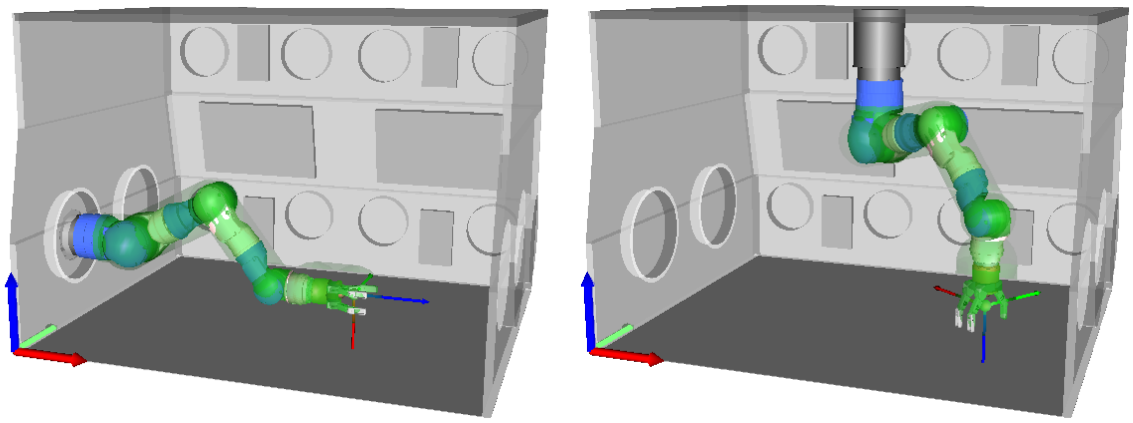


Figure 5-8: Two base locations for the Motoman SIA-10 manipulator in a glovebox environment. The manipulator is making a side grasp on the left and a top grasp on the right.

In a glovebox, different mounting locations/orientations are possible: the floor, ceiling, or side walls. We select base frames that would most likely be used in an actual glovebox. The manipulator was first mounted in the side access port with rotations of 45º and 90º from the vertical, where a positive rotation is CCW about the positive y-axis. In the global frame, this corresponds to a base frame location of (0 mm, 419.1 mm, 317.5 mm). For each rotation, we generated task planes at 101 mm and 200 mm. These heights were chosen based on our manipulator obstacle model and the assumption that most manipulation will occur near the glovebox floor.

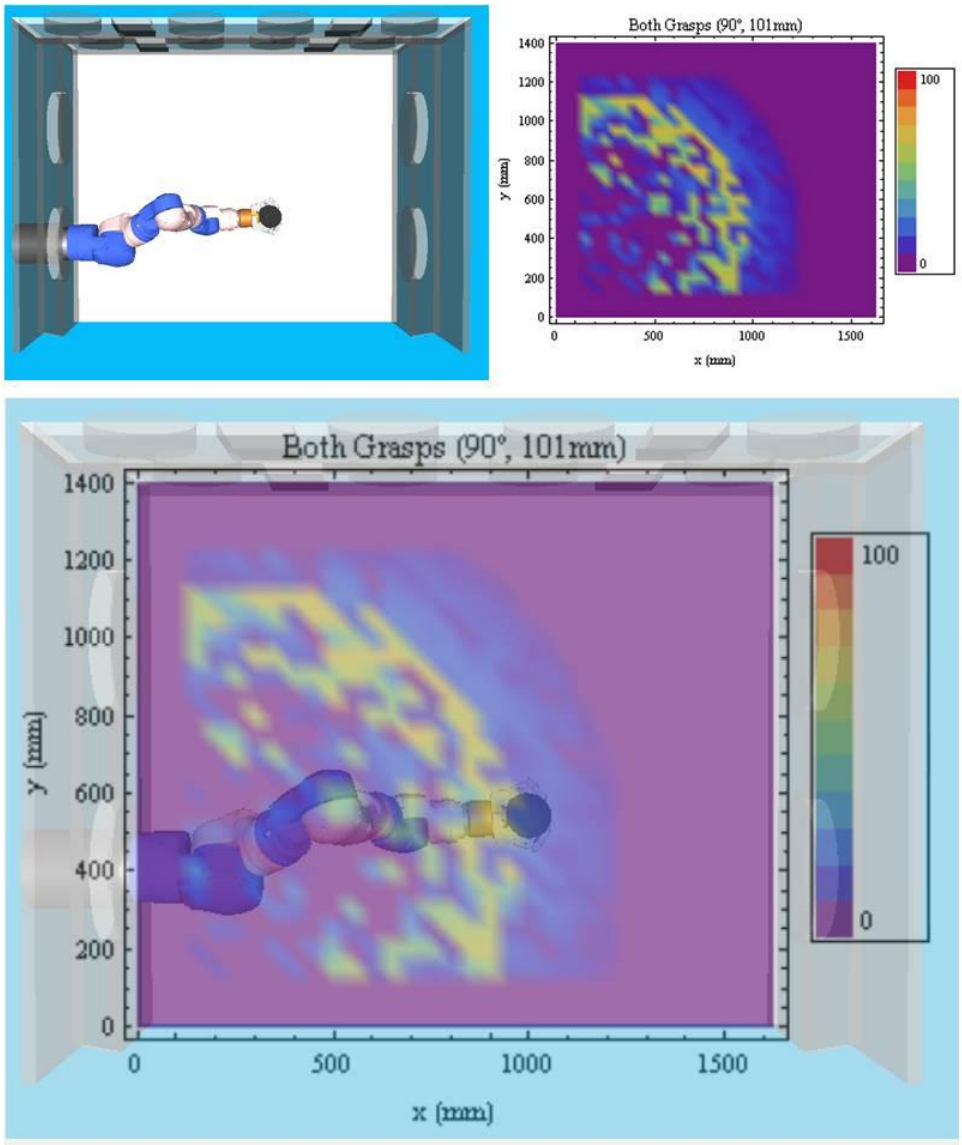


Figure 5-9: A manipulator mounted in a glovebox side access port (top left) and the task plane that corresponds to this manipulator base location (top right). The task plane is then overlaid on the actual glovebox configuration and workspace it represents (bottom).

For a manipulator mounted in the center of the ceiling, we located the base frame at three different distances between the base frame and the task plane: 743 mm, 842 mm, and 942 mm. The x-location of the base was 812.8 mm and the y-location was 685.8 mm. A z-location (height) of 943 mm was used to generate 101 mm and 200 mm task

planes, and a 101 mm plane was generated for a height of 1043 mm. The task planes generated included side, top, and both/total (side and top) grasps.

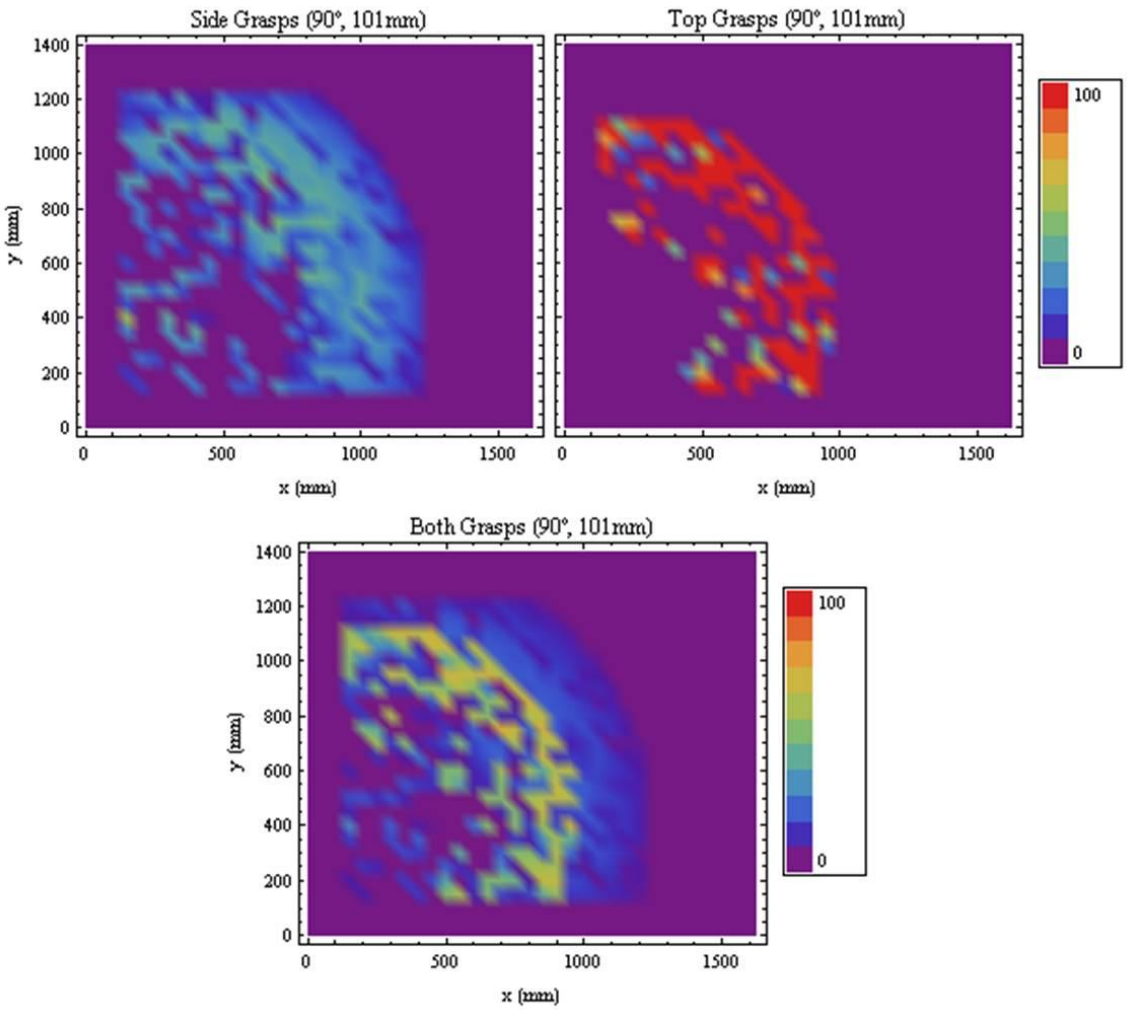


Figure 5-10: Task planes for the Motoman arm base frame located at (0 mm, 419.1 mm, 317.5 mm) and rotated 90º about the global y-axis. The plane height is 101 mm and there is a plot for side, top, and both grasps.

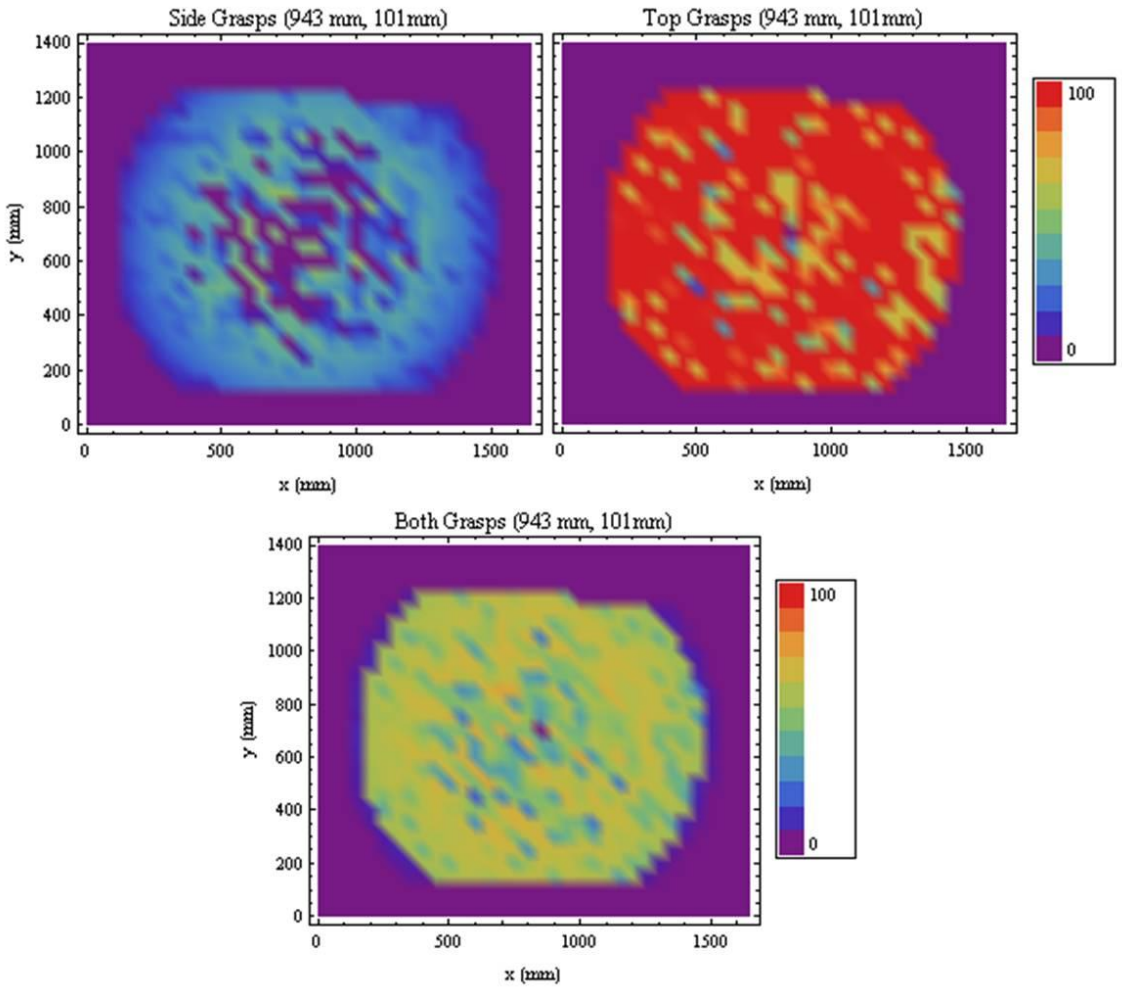


Figure 5-11: Task planes for the Motoman arm base frame located at (812.8 mm, 685.8 mm, 943 mm) and rotated 180º about the global y-axis. The plane height is 101 mm and there is a plot for side, top, and both grasps.

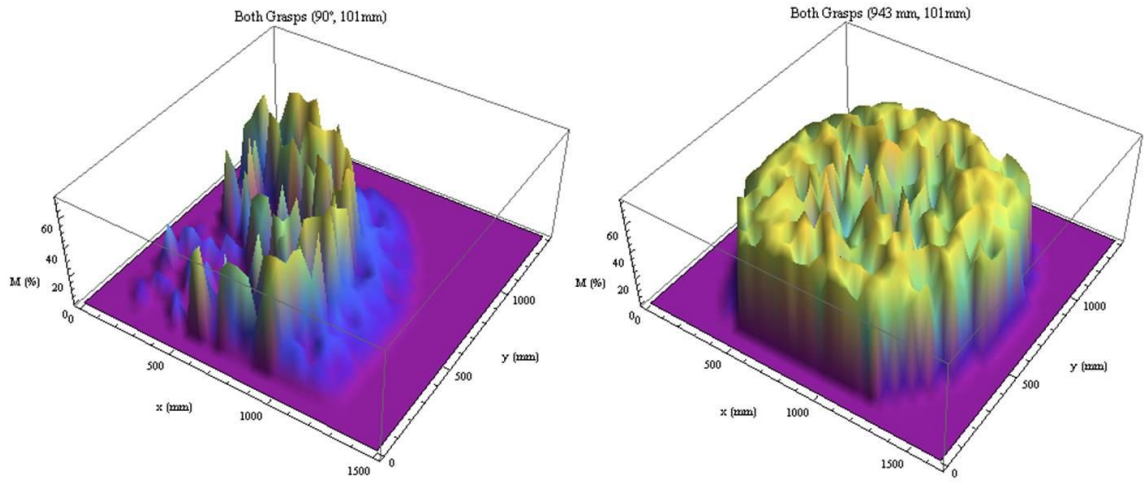


Figure 5-12: Three dimensional plots of the task planes in Figure 5-10 and Figure 5-11 combining both grasps.

We calculated the AMI to determine the best base location from these trials. We first defined a representative area of a glovebox worker's workspace in a set of gloveports: 700 mm by 500 mm. Then we searched the task plane for the location that gave the best AMI for this area. Higher AMI for the representative workspace corresponds to greater manipulability for the type of grasp. Thus, comparing the quality of different base positions reduces to comparing the AMI values for planes generated for those base positions. The results of these tests are in Table 5-1.

Table 5-1: Best location search for a representative workspace in a glovebox

Test Configuration	Grasps	AMI (%)	# Spheres Inside Target	Best Location (mm)			Target Rotation ϕ (deg)
				X	Y	Z	
Access port: 90° Plane height: 101mm	Side	24.1726	141	850	750	101	120
	Top	40.199	141	650	750	101	150
	Both	30.3044	141	650	750	101	150
Access port: 90° Plane height: 200mm	Side	25	141	900	750	200	120
	Top	33.9342	141	600	700	200	120
	Both	26.315	141	650	700	200	120
Access port: 45° Plane height: 101mm	Side	18.219	153	750	500	101	270
	Top	67.0016	141	500	750	101	150
	Both	40.8146	141	600	600	101	120
Access port: 45° Plane height: 200mm	Side	22.8329	141	550	800	200	180
	Top	55.5162	141	650	550	200	120
	Both	38.4062	141	550	750	200	150
Ceiling mount: 943mm Plane height: 101mm	Side	26.5465	141	500	800	101	60
	Top	93.1792	146	900	950	101	180
	Both	59.1562	146	900	950	101	180
Ceiling mount: 943mm Plane height: 200mm	Side	29.4045	146	1050	950	200	180
	Top	63.1895	141	400	550	200	120
	Both	45.5213	158	400	850	200	90
Ceiling mount: 1043mm Plane height: 101mm	Side	17.1633	165	800	750	101	0
	Top	91.6286	146	750	450	101	180
	Both	54.0002	146	750	450	101	180

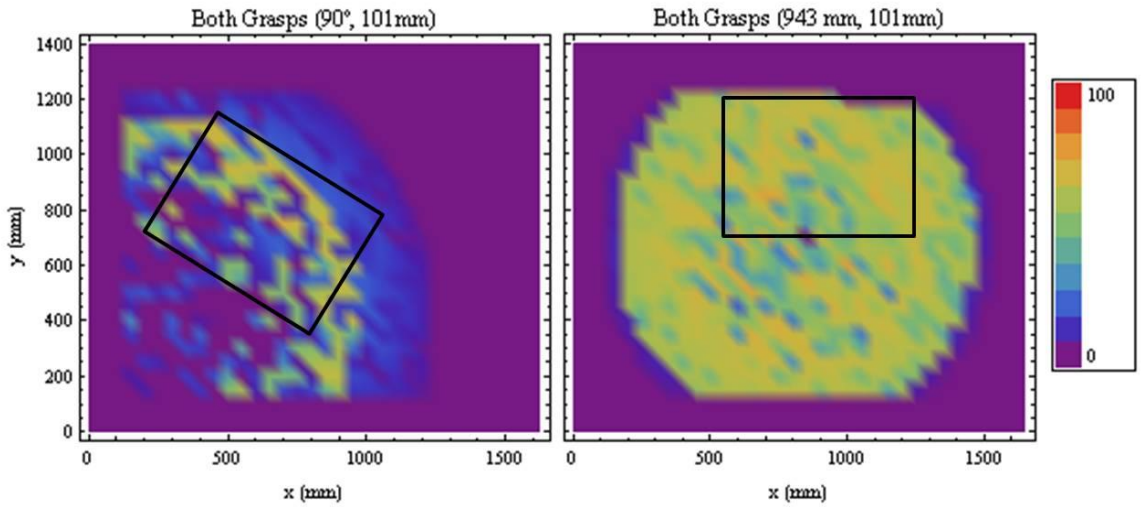


Figure 5-13: The best target location in glovebox task planes for a representative worker area of 700 mm by 500 mm. Two of the twenty-one trials in Table 5-1 are shown.

We draw a number of conclusions from this data. For mounting in the side access port, side grasps are more easily performed when the arm is mounted at 90º. Top grasps are easier when the arm is mounted at 45º. This can be attributed to the higher z-location of the first pitch joint for 45º mounting. For both grasps, in general, we see that the manipulability index for top grasps at a grid point has the greatest influence on the index for both grasps. As previously discussed, usually all or none top grasps are possible. Thus, for the both index, either the top index is low and the total index is closer to the side index, or the top index is high and the total index is much higher than the side index. This suggests that in some situations the total index may not be the most robust for comparison.

For ceiling mounts, top grasps are performed with greater ease than side grasps, as expected. The greater the distance between the base frame and task plane, the harder it is for the manipulator to perform side grasps. Again, the high values of the top index dominate the total index. When comparing side mounts to ceiling mounts, it is more

appropriate to compare the side and top indexes independently. The side indexes are relatively similar, but the top indexes for the ceiling mounts are much higher. Therefore, a ceiling mount may be preferred, and it also has the added value of covering a larger area with greater manipulability, as seen in Figure 5-11.

Some of these conclusions are intuitive given the dimensions of the arm and its mounting locations, but now the task planes and AMI give a quantitative measure and a visual that confirm our expectations. In general, other factors would also be weighed in the base location decision. For example, ceiling mount implementation could be very difficult or containment/contamination issues may be alleviated by a certain mounting location.

These types of results can also direct additional base position searches if a satisfactory location is not initially found. Based on the results and analysis above, we suggest another mounting location if a ceiling mount is not possible. For ceiling mounts, the best top index was found for a base z-location of 943 mm. This corresponds to a height of 678 mm for the location of the first pitch joint. Thus, mounting the arm at 90° on a side wall at a height of 678 mm gives the same location for the first pitch joint and should improve top grasping. Additionally, for ceiling mounts, the side grasp index was highest for a distance of 743 mm between the base frame and task plane. For side mounting at a height of 678 mm, a task plane at 101 mm rather than 200 mm would give a displacement closer to what we desire. The task planes for this new configuration are shown in Figure 5-14.

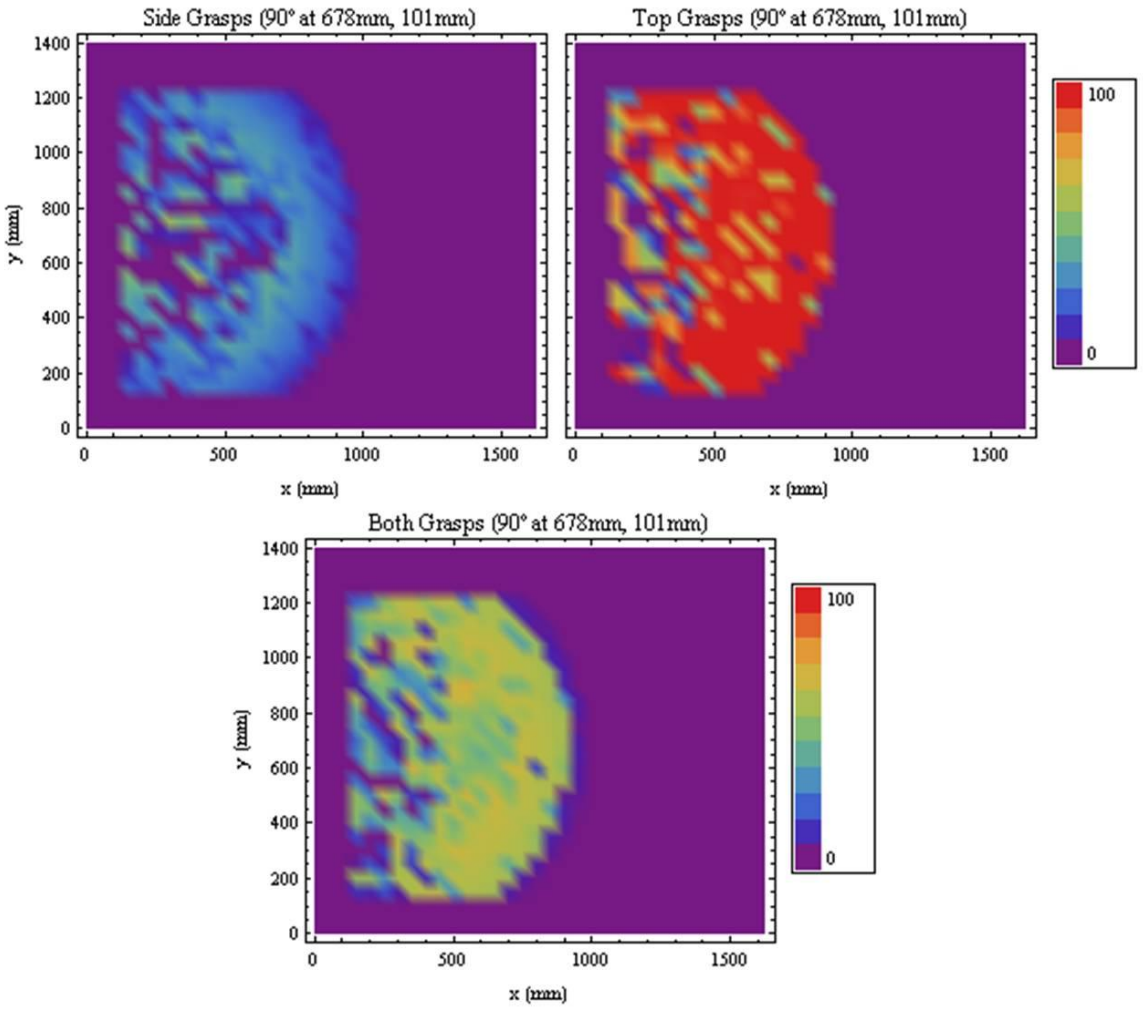


Figure 5-14: Task planes for the base location developed from previous tests.

In the planes for this new system configuration, we see a side index in the range of those previously found for both mounting types. Compared to other side mounts, there is an improved top index and also a larger, more consistent workspace area where higher manipulability is possible.

Table 5-2: Best location search in the glovebox for the derived base location

Test Configuration	Grasps	AMI (%)	# Spheres Inside Target	Best Location (mm)			ϕ (deg)
				X	Y	Z	
Side wall: 90°	Side	22.1533	141	600	700	101	120
Height: 678 mm	Top	86.7021	141	600	550	101	60
Plane height: 101mm	Both	54.0829	141	600	550	101	60

If we select this base location, we can then locate targets about the base. In the previous tests, we used a representative workspace to define an area where undetermined tasks could be performed. These tests calculated the AMI to compare planes and find the best location for the worker's representative workspace. We can also define specific targets and find their best position for manipulation relative to the base frame. This accomplishes the same goal as Abdel-Malek [2004].

Consider the layout for the MOX glovebox described in Section 3.1.3 and seen in Figure 3-3. This layout was found by trial-and-error: we first chose the base and target locations and then planned and executed motions, checking for inverse errors and joint limit exceptions. Instead, task planes can be used to solve this location problem. We take some of the targets around the right manipulator and put them in the glovebox with the Motoman arm. Assuming these targets are in a favorable, fixed configuration, we perform a best location search.

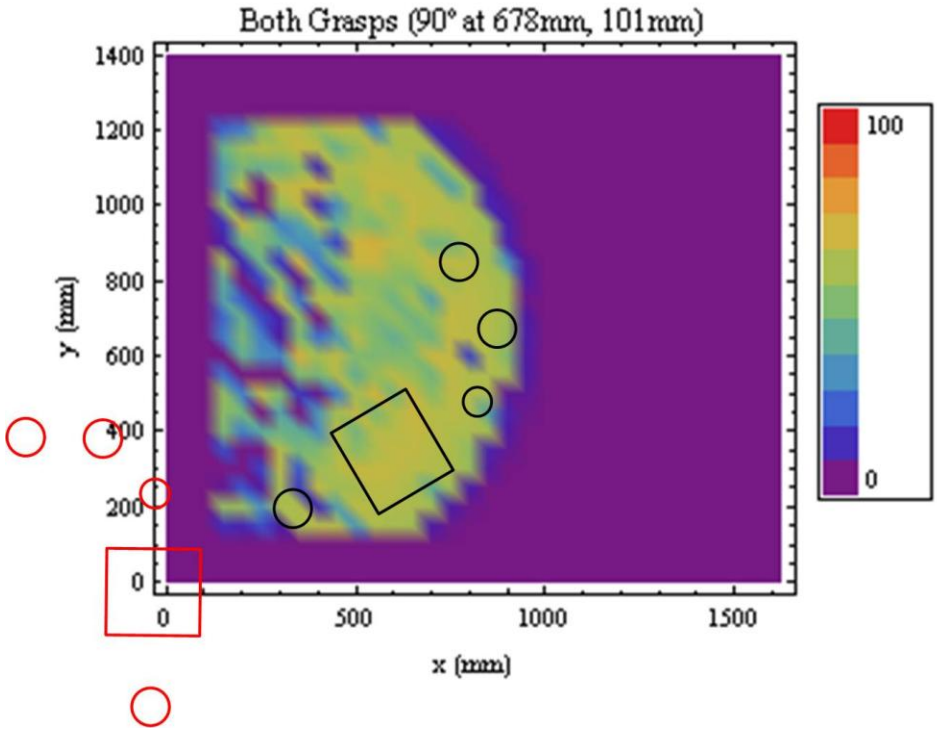


Figure 5-15: Best target location search for a group of targets derived from MOX glovebox targets. Cans are represented as circles and the scale is represented by a rectangle. The target group in red shows the initial trial position of the targets. The final (and best) location is shown in black.

The search finds a location that positions the targets in a region of high manipulability. The initial target group is constructed with a local axis at the origin and coincident with the center of the rectangle target. The search moves the local axis from the grid point at (0 mm, 0 mm) to one at (600 mm, 350 mm) and rotates the targets by 300º CCW about this point. In the glovebox task planes, there is often an arc surrounded by higher manipulability resulting from the reach of the robot. The group of targets is rotated about the z-axis to better align the targets with this arc. At this position, there are 37 grid points within the targets with an AMI of 62.4%.

Using task planes for configuration management has many benefits. Trial-and-error methods are very time consuming for an operator, but the MTPA greatly reduces the time required to configure a new system. Even if the operator finds a configuration by trial-and-error, how "good" this configuration is, how it compares to other possible configurations, and how adaptable it will be to adding new tasks is unknown. The MTPA provides both visualizations and quantitative metrics for operator comparison.

5.3.1.2 Manipulator Evaluation

As mentioned, task planes reveal information about the capabilities of a manipulator. Many times, the qualities in a task plane can be linked directly to its physical geometry. Due to this, different manipulators generate different task planes that can then be compared to select a manipulator based on the needs of the task.

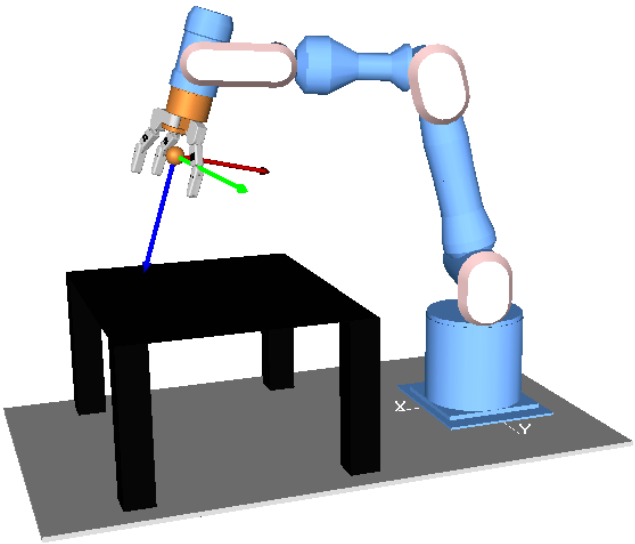


Figure 5-16: Experimental setup for manipulation on a table. The Mitsubishi PA7C robot is shown.

For this test, we examine manipulation on a table top using three different manipulators: the PowerCube LWA3, Mitsubishi PA7C, and Motoman SIA10. Each manipulator has dimensions, limits, etc, characteristic of those from a commercial build, and we assume that these attributes are fixed. For the task, we assume a fixed base position and workspace area relative to the base. The base location for each robot sets the axis of the first pitch joint at a location of (0 mm, 0 mm, 317 mm). Table top is at (600 mm, 0 mm) relative to the base frame with dimensions of 500 mm by 600 mm (x by y). Task planes were generated for a rectangle with the same center and dimensions of the tabletop at a height of 101 mm above the table surface. The resulting task planes for each manipulator are below.

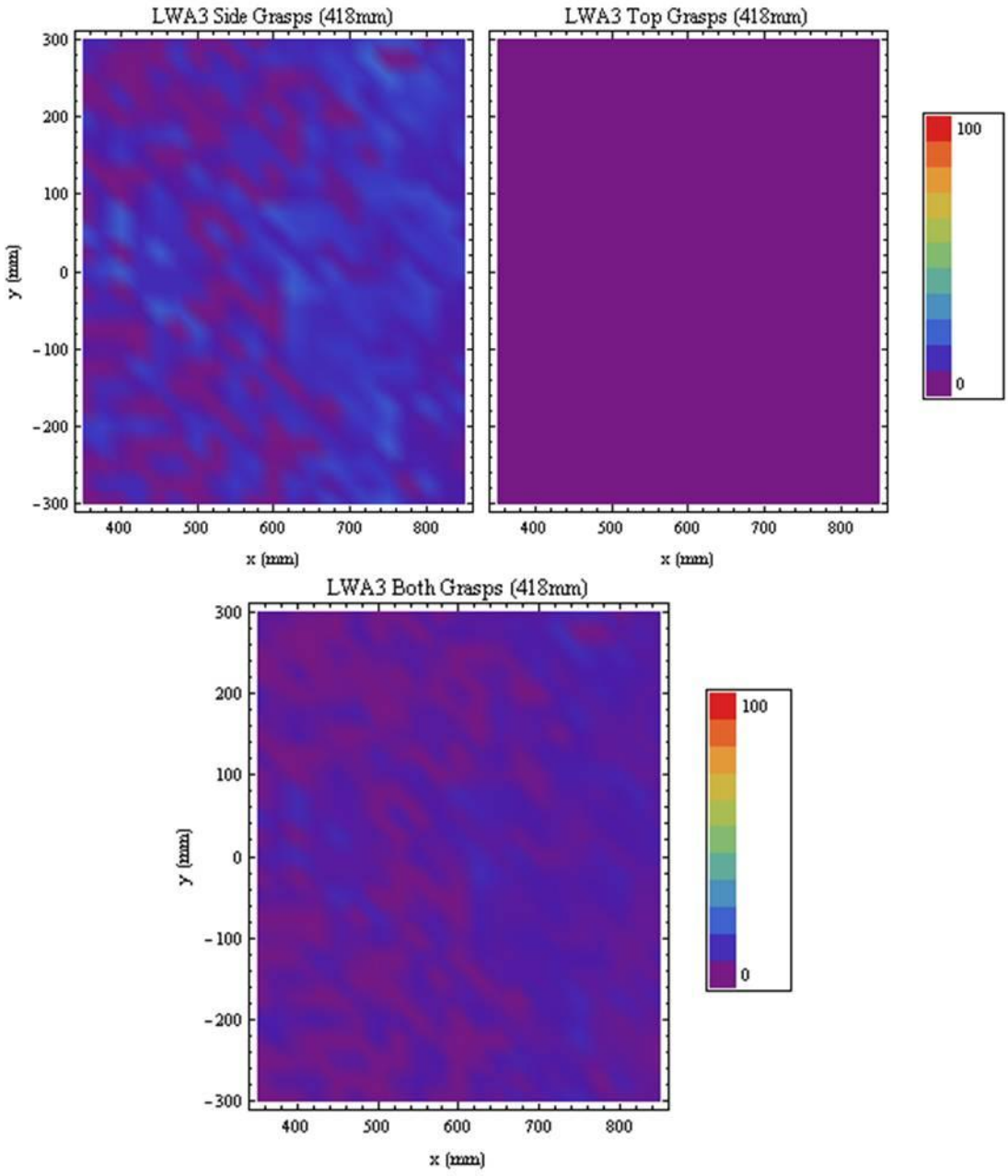


Figure 5-17: Task planes for table manipulation by LWA3 arm.

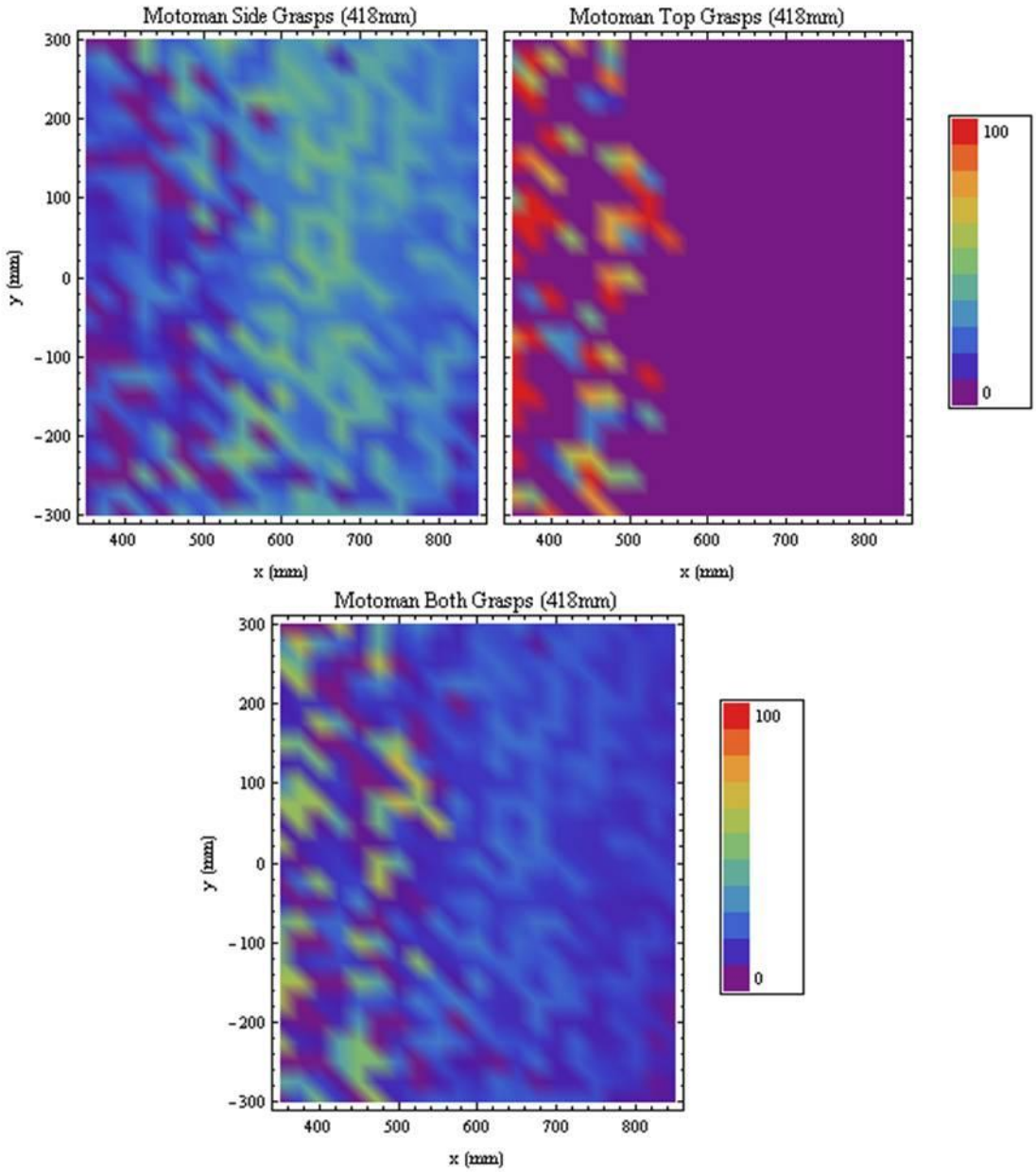


Figure 5-18: Task planes for table manipulation by Motoman arm.

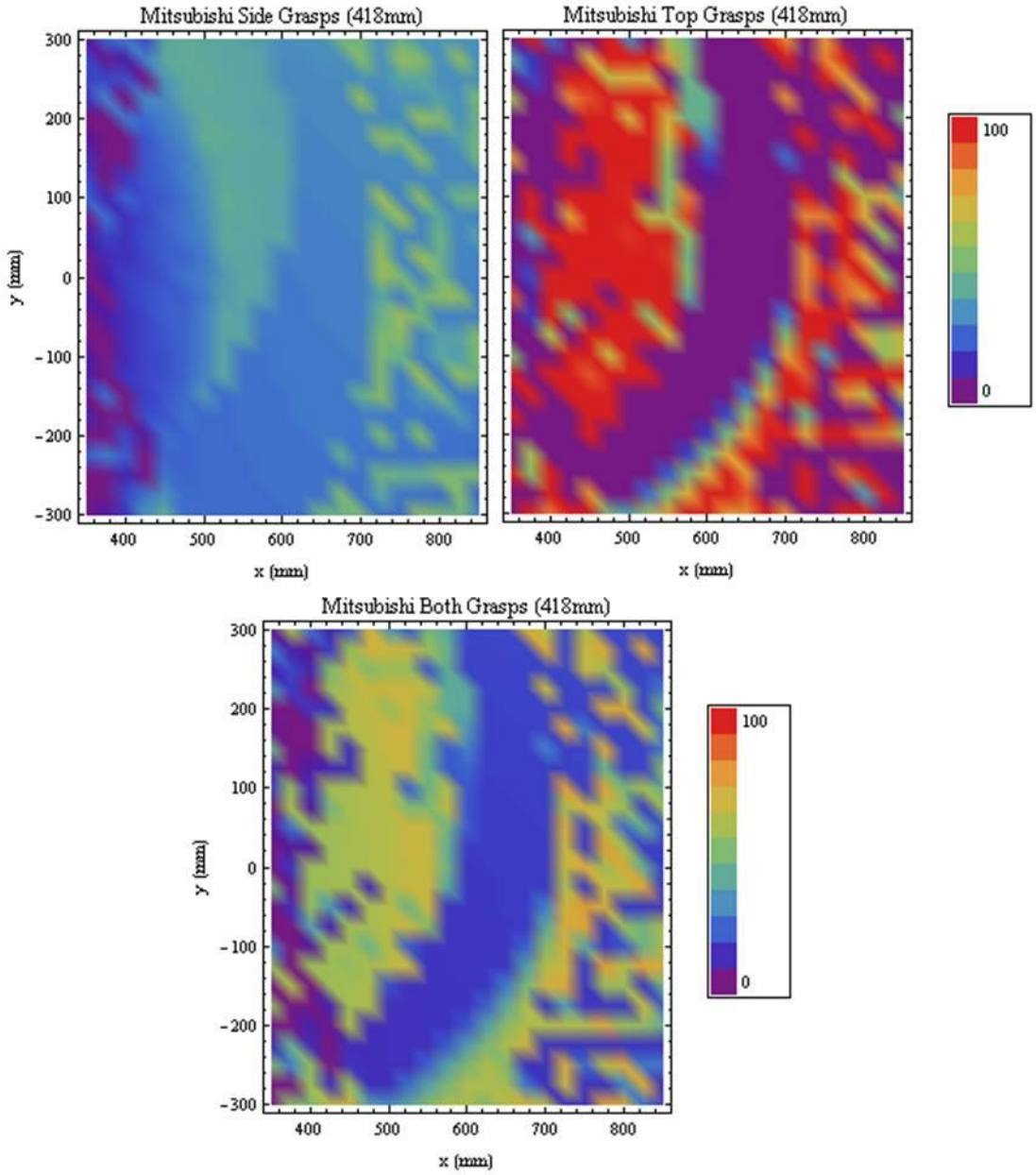


Figure 5-19: Task planes for table manipulation by Mitsubishi arm.

The task planes reveal the capability of each manipulator for this basic task. The LWA3 arm can execute side grasps throughout the table area but can do so more easily at table locations further away from the base. Also, the arm cannot execute any top grasps.

Physically, this makes sense because of the shorter reach of the arm and the long distance between the last joint and the tool point. The Motoman arm displays similar traits as the LWA3 arm for side grasps but with increased manipulability. The Motoman can execute top grasps, however, there is clearly a boundary where top grasps cannot be performed. The Mitsubishi robot can execute side grasps with about the same manipulability as the Motoman arm, and can perform top grasps for a larger portion of the workspace area. This is reasonable considering that top grasp capability is aided by a longer link length between the 4^{th} and 6^{th} joints combined with a short distance between the last joint and tool point – something the LWA3 arm lacked.

Based on this analysis, the Mitsubishi robot is most likely the best selection given specific tasks that are undefined and assuming that performance and other constraints are met. There appears to be a section of reduced manipulability in the Mitsubishi task planes between areas of higher manipulability. The arm is not at a singularity, so the cause appears to be the $\boldsymbol{\varphi}_i$ found by the IK algorithm in this area of the plane. For some $\boldsymbol{u}_{des}$, the IK gives a solution but joint position limits are exceeded or a collision occurs. At some points, for instance, half of the side grasps give non-colliding configurations with the 4^{th} joint above the table top, but the other side grasp configurations collide because the $\boldsymbol{\varphi}_i$ chosen have the 4^{th} joint below the table surface. Changes in the solution branches used at a point or between adjacent points also appear to effect top grasps. Better techniques of exploiting the manipulator redundancy could help resolve this issue.

Although a simple test, this shows some principles that are intuitively known concerning how robot geometry affects capability. The influence of other parameters like joint limits could also be analyzed as well as joint failure. With any evaluation, the AMI of the planes and other criteria could also be calculated and compared to add to the

analysis, but the purpose of this test was to demonstrate that task planes can visually show differences in capabilities and aid in manipulator evaluation/selection.

5.3.1.3 Placement of Mobile Manipulator

This is a task plane application to a demonstration conducted at the RRG. The objective was to position the base frame of a manipulator located on a mobile platform so that the arm can open a cabinet door and extract a canister that is selected only after opening the door. Both tasks must ideally be completed without repositioning the platform.

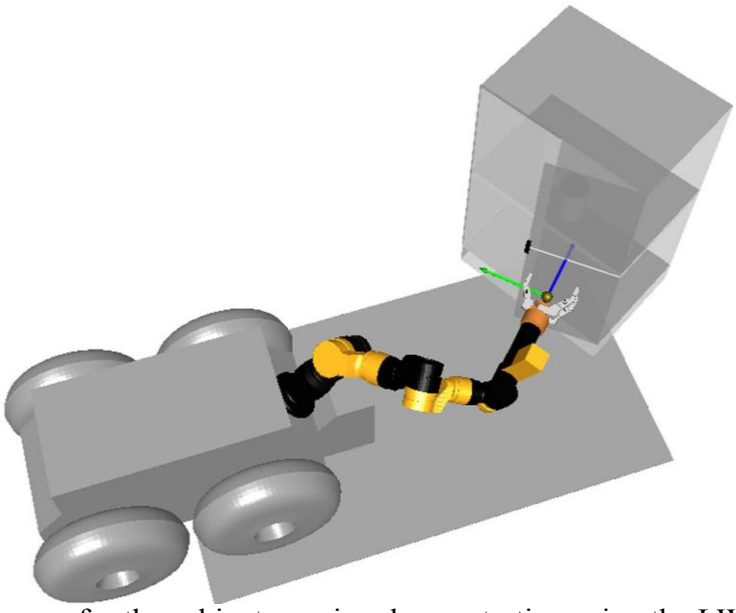


Figure 5-20: Setup up for the cabinet opening demonstration using the LWA3 arm.

Initial identification of a valid base location was done in simulation using trial-and-error. Locations were found in simulation for which the door handle could be grasped, the door opened, and a canister reached. However, some manipulator motions were not simulated. Thus, when the entire task was tried using hardware, the manipulator

could not perform it since the paths between target points could not be traversed due to kinematic limitations (primarily exceeding the joint position limits).

From trial analysis, the inverse kinematic algorithm used with the LWA3 arm appeared to be limiting the arm's capabilities. This IK algorithm sets the first joint of the manipulator then analytically solves for the last six joints for a closed-form solution. Task planes can be used to show differences, if any, between IK algorithms. At the same time, the planes give a more robust location of the optimal base location as compared to previous trial-and-error attempts.

The use of task planes, however, is limited in this application. They cannot be used to model the trajectories needed to accomplish the entire task. These trajectories are not confined to a plane and hand orientations of any type could be used. Nevertheless, the task plane should reveal information about the closed-form IK algorithm used and how it compares to a redundant algorithm that utilizes a JRA criterion.

The task planes were generated at a height of 612 mm, which is the approximate height of the cabinet handle. The center height of the can in the cabinet is a couple cm from the handle height. From Figure 5-21, it is apparent that the two IK algorithms show different capabilities in the workspace. The closed-form IK shows a smaller radius of valid grasps about the location of the first pitch joint, which is approximately located at (648.6 mm, 0 mm) in the plots. The JRA IK algorithm finds more valid grasps in a larger area of the workspace. We also see a steep drop-off in manipulability that occurs when moving from the highest areas of manipulability toward the base of the robot. This may explain some of the difficulty in repeatability found in our previous trials, as even a small movement in position may lead to large changes in capability.

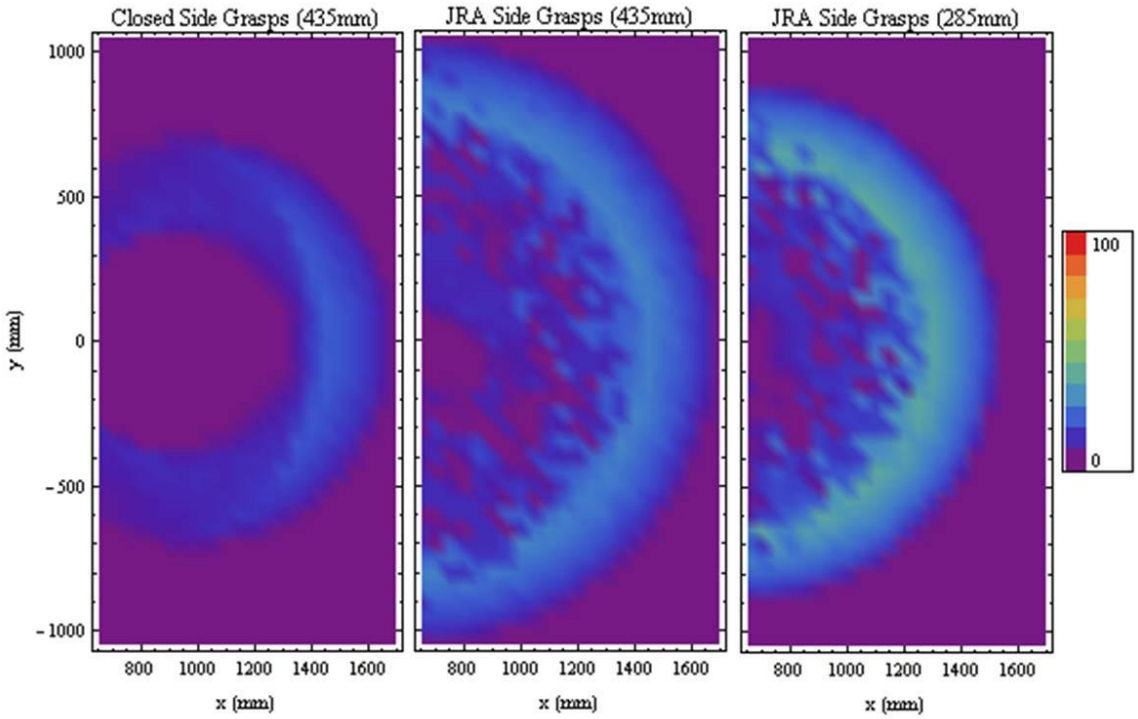


Figure 5-21: Comparison of task planes for the workspace of the LWA3 arm mounted at 45º on a mobile platform. The first two planes on the left are for different inverse kinematic algorithms: a closed-form solution and one that incorporates the Joint Range Availability criterion. The plane on the far right is generated with a 150 mm shorter tool point and uses the JRA IK.

We also performed a best target search to compare the AMI for a representative task execution area. Two representative work areas were selected that contain the objects to manipulate and the possible location of trajectories. These areas were a 200 mm square and a 400 mm square. The JRA IK task plane was found to have a higher AMI for both areas, as seen below in Table 5-3. Repeating the demonstration using the JRA IK should alleviate some of the problems encountered, leading to the successful completion of the entire task. This search also shows the preference of the arm to manipulate targets in areas to the left or right of the straight forward direction/x-axis, and that the best target locations are rotated to align themselves with the arc of higher manipulability.

Table 5-3: Comparison of task plane AMI for LWA3 IK methods and tool points

LWA3 Test IK	Target Area (mm^2)	AMI (%)	# Spheres Inside	X (mm)	Y (mm)	Z (mm)	ϕ (deg)
Closed Form	200 x 200	16.4216	17	1500	0	612	30
JRA (tp = 435 mm)		23.0159	21	1350	400	612	180
JRA (tp = 285 mm)		31.3725	17	1250	-250	612	30
Closed Form	400 x 400	10.9829	65	1400	250	612	60
JRA (tp = 435 mm)		18.8034	65	1400	150	612	60
JRA (tp = 285 mm)		25.0427	65	1300	-50	612	30

Table 5-3 also lists AMI information for a third task plane generated using the JRA IK but with a tool point 150 mm shorter in the z-direction than the previous JRA test. During testing, the long length of the tool point appeared to hinder the manipulability of the arm in addition to the closed-form IK. We see an improvement in the AMI for this shorter tool point. Physically, this tool point cannot be implemented due to limits in the hardware. However, from this test, we see that task planes can visualize the changes in capabilities that occur from changes in robot geometry.

5.3.2 Application to Motion Planning

Our main goal is to demonstrate that we can use task planes to motion plan using a skeleton/graph approach. In general, we select an area of good manipulability from a previously generated task plane to plan our trajectories. Motions are conducted in the plane and the hand orientation is fixed for either a side or top grasp, with a free rotation θ_f of $\boldsymbol{u}_{des}$ about the global z-axis. After the A* algorithm generates a path of task plane grid points, we use the directional information stored in the plane to specify $\boldsymbol{u}_{des}$ along the trajectory. These $\boldsymbol{u}_{des}$ are specified at each grid point in the path, and small motion plans are made between these way-points to generate the full trajectory.

We also place obstacles in the environment, generating planes both without and with these obstacles. To analyze the influence of obstacles, we examine the AMI of the path found by the A* algorithm and look qualitatively at the motions used to avoid the obstacles.

5.3.2.1 Planning Side Grasp Trajectories

To demonstrate the motion planning capability for side grasps, we selected an area from the 101 mm task plane of the Motoman robot mounted at 90º in a side access port (Figure 5-22). This region contained a large amount of relatively high manipulability. A task plane with 25 mm grid spacing was generated. We then populated the task plane with four obstacles (see Table 5-4) and re-generated the task plane. These obstacles were modeled as `Cylisphere` and `Plane Primitives` in the OSCAR workcell XML file.

Table 5-4: Obstacles for side grasp motion planning

Object	Radius (mm)	Height (mm)	Position of center (mm)		
			X	Y	Z
Can	52.4	136.5	1150	500	68.3
Can	52.4	136.5	700	150	68.3

Object	Length (mm)	Height (mm)	Thickness (mm)	Position of center (mm)		
				X	Y	Z
Plane	200	300	20	600	1100	150
Plane	200	300	20	975	1100	150

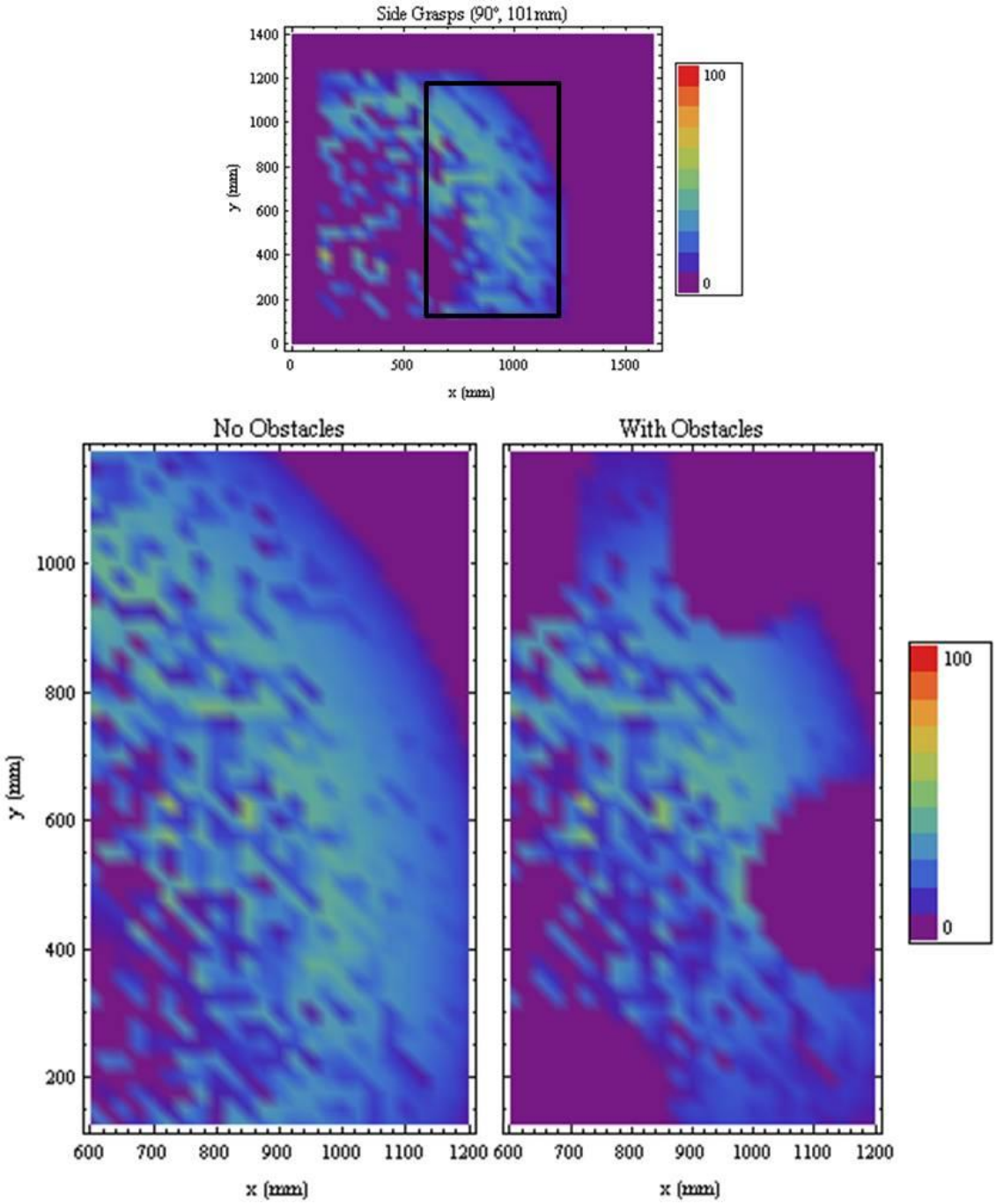


Figure 5-22: A region of relatively consistent manipulability (black rectangle) is chosen from a previously computed task plane. This region is used to generate a new, finer task plane both without and with obstacles.

The obstacle task plane clearly shows the presence of these objects. At a grid point, any trial grasp is eliminated if the collision model of the manipulator intersects the obstacle model of an object. Thus, no grasps are possible at grid points that are within the volume of the object. Additionally, grid points near the objects may have a reduction in the number of grasps previously attainable because certain grasps now lead to configurations that collide with an obstacle. In the obstacle task plane, this is seen in the regions between the two cans and between the two planes. The colors in this region are darker when compared to the original task plane, meaning the manipulability indexes of those points have decreased.

Some snapshots of a trial motion plan in the presence of these obstacles are seen in Figure 5-23. These shots show general trends in certain workspace regions. The manipulator starts its motion between two cylindrical containers and ends it motion with its EEF between the two planes. This path could represent the task of retrieving a can from among others and moving it from one side of the glovebox into a cabinet or box on the other side.

The effect of the task plane data on the trajectory can be seen in Figure 5-23. The hand begins in the configuration seen in Figure 5-23A and moves up and to the left toward the right can (Figure 5-23B). When close to this can, the selected $\boldsymbol{v}_{gd}$ rotate the hand CW about the global z-axis to guide it around the edge of the can and move the arm away from the left can (Figure 5-23C). When the hand clears the left can, it attains $\boldsymbol{v}_{gd}$ that begin to rotate it CCW and direct it toward the plane objects (Figure 5-23D). As the hand approaches the right plane, it is eventually shifted to the left in order to execute its motion between the planes and to the final location (Figure 5-23E and Figure 5-23F).

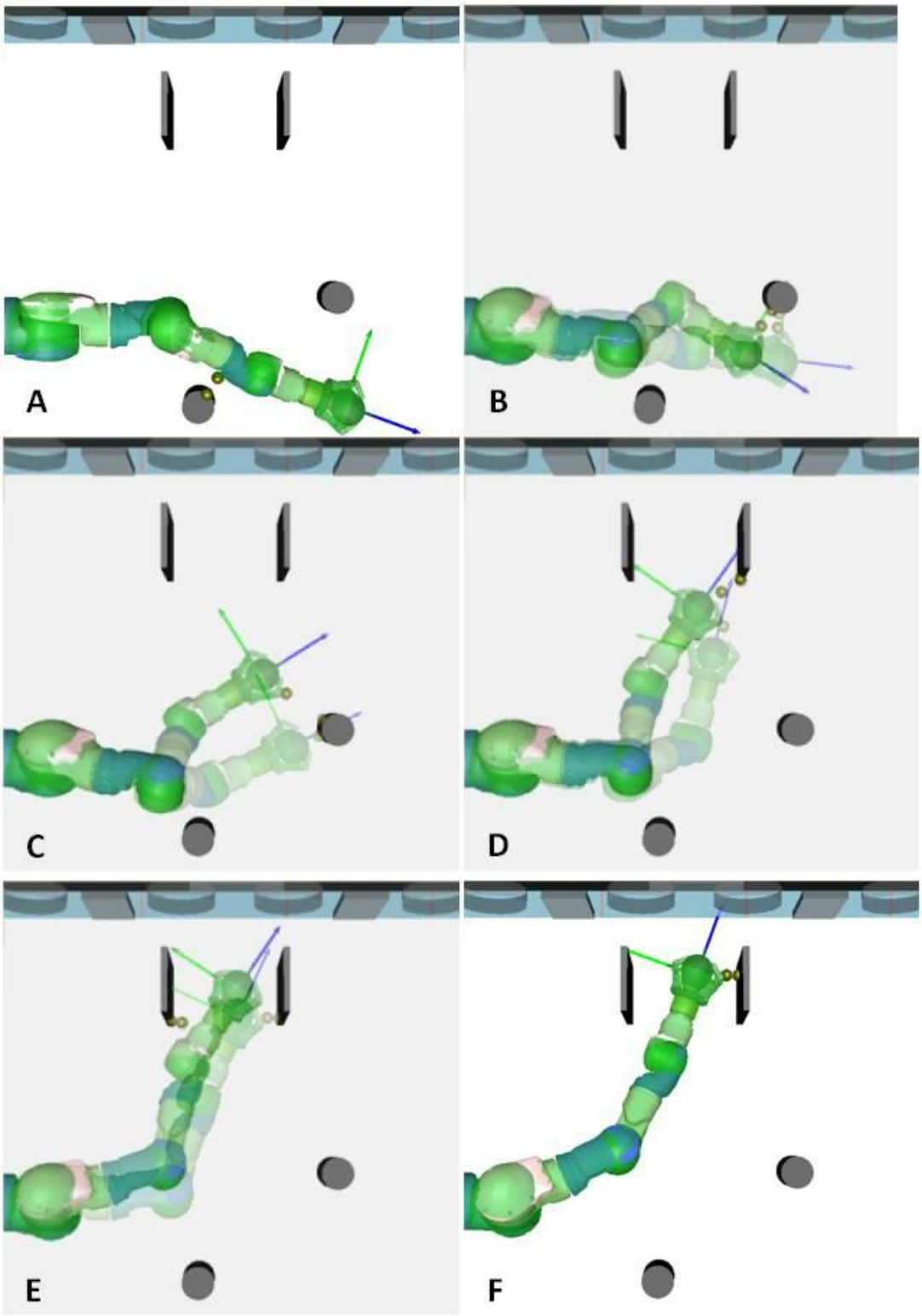


Figure 5-23: Snapshots of the side motion plan for the Motoman arm.

For the starting and ending points above, a path was found both with and without obstacles (Table 5-5). The A* algorithm finds the shortest cost (highest AMI) path from the starting grid point to the end grid point. Thus, the EEF's path lacks unnecessary motions and drives it toward the end point from the beginning of the motion. The AMI for the path without obstacles is higher than the path with obstacles. The AMI of both paths is higher than the respective AMI for the entire plane.

Table 5-5: AMI information for path test with side grasps

	Start Point		End Point		# Spheres		Path	Plane
Side Grasp Path Test	X	Y	X	Y	Inside	Sum	AMI (%)	AMI (%)
No Obstacles	41	1	1	15	55	1691.67	30.8	18.8
With Obstacles	41	1	1	15	55	1401.39	25.5	11.0

Each path is shown in the directional plots of Figure 5-24 and Figure 5-25. The execution of each path was successful in simulation. In the directional plots, the (0, 0) node is in the top right corner, with the first node value increasing to the left across the row, and the second node value increasing down a column. Figure 5-25 shows the approximate grasping directions $\boldsymbol{v}_{gd}$ (green arrows) selected for each configuration in Figure 5-23. Again, $\boldsymbol{v}_{gd}$ is used to select θ_f which determines the $\boldsymbol{u}_{des}$ used at that grid point along the path. The chosen $\boldsymbol{v}_{gd}$ are the vectors in the middle of the longest continuum of valid $\boldsymbol{v}_{gd}$ (blue arcs), i.e. they bisect the longest blue arc.

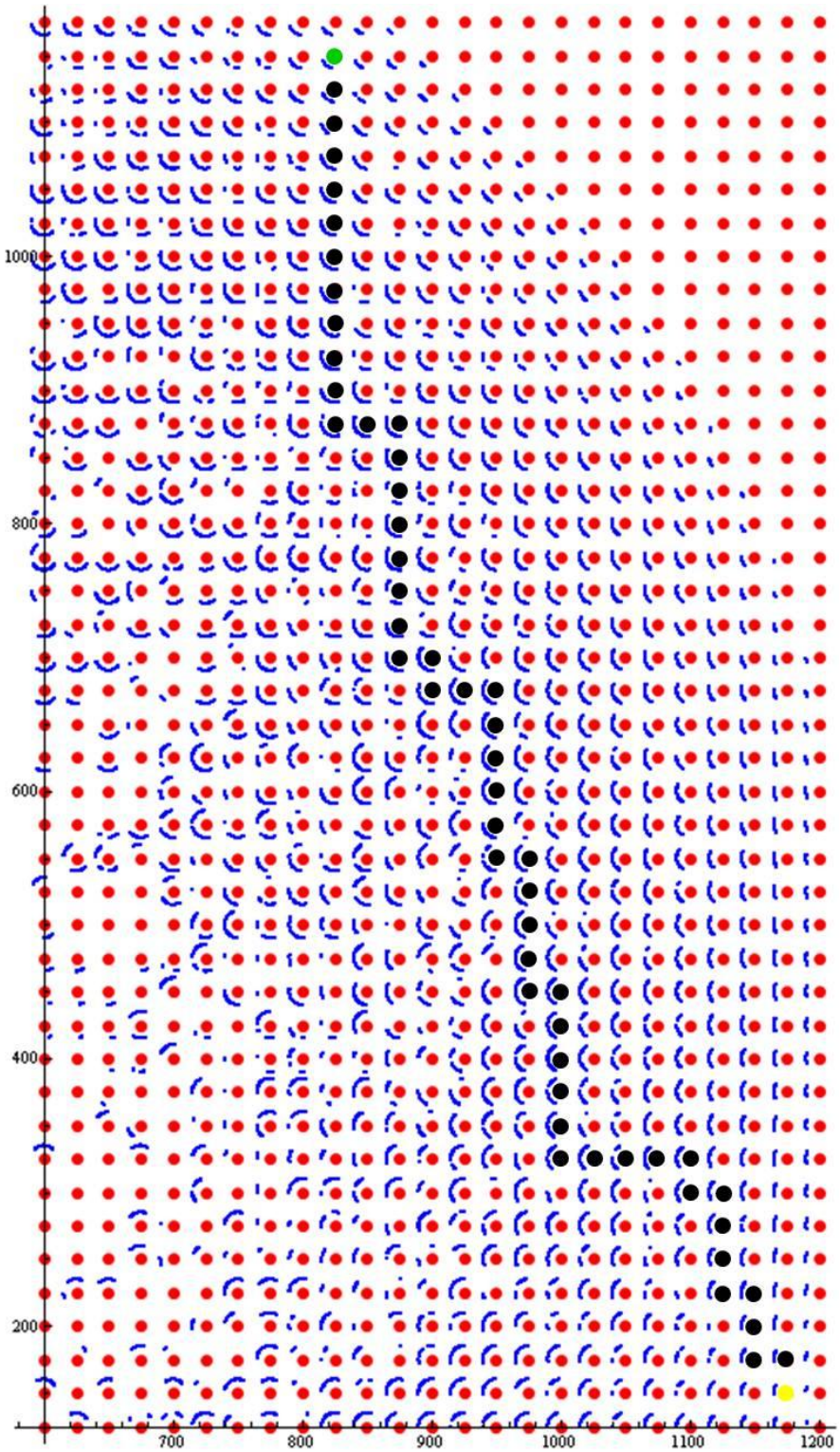


Figure 5-24: Directional plot and path for the case with no obstacles. The starting point is in yellow and the end point is in green.

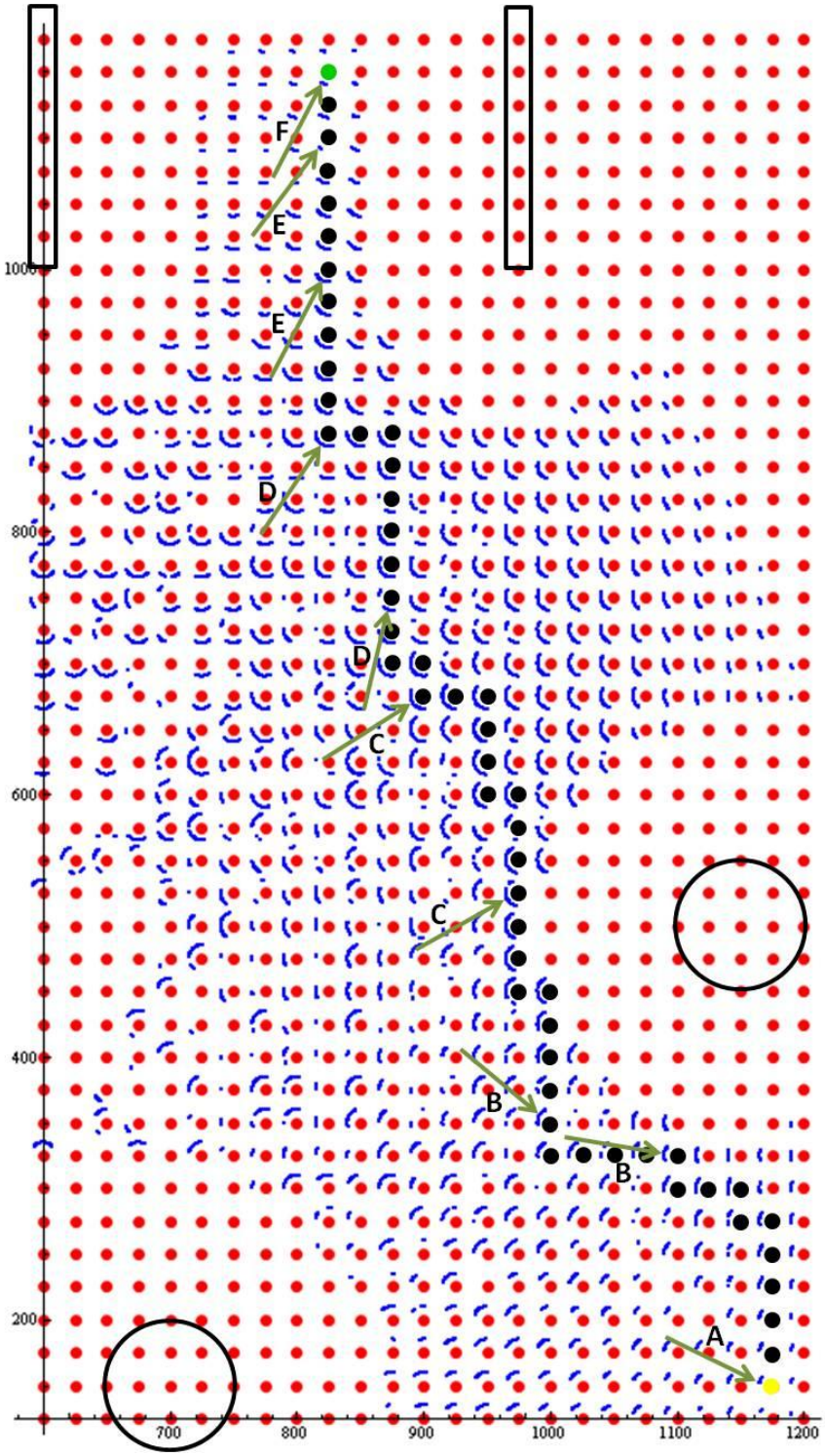


Figure 5-25: Directional plot and path for the case with obstacles. The starting point is in yellow, the end point is in green, and the approximate location of obstacles is outlined in black. The approximate $\boldsymbol{v}_{gd}$ used for the configurations in Figure 5-23 are in green.

The directional plots provide additional information about the effects of the obstacles. In the obstacle planes, many of the valid sectors (blue arcs) are shorter in length meaning the manipulability index at those points has been decreased (i.e. less valid configurations are possible). As expected, farther from the robot, grasps on the other side of the grid point are often unreachable (i.e. the object can only be grasped from the front as viewed from the base frame). However, nearer to the base frame the robot is able to execute grasps that grab the can closer to its backside. Closer in toward the robot, there are also areas with more variation – sometimes adjacent points have many valid points in common, other times they do not. For the more consistent regions of the plane, the valid $\boldsymbol{v}_{gd}$ appear to be biased according to the grid point's position relative to the location of the first pitch joint. In this plane, this joint is at (265 mm, 419.1 mm). Drawing a vector from this point to a grid point of interest intersects the blue arc around the point.

The obstacle directional plots also show how the $\boldsymbol{v}_{gd}$ selected for a configuration changes with the incorporation of obstacles. The obstacles can reduce the valid $\boldsymbol{v}_{gd}$ at a point, changing the position and/or shape of the longest continuum of valid $\boldsymbol{v}_{gd}$ and thus the $\boldsymbol{v}_{gd}$ in the middle of the continuum (which is chosen for the configuration). Essentially, the obstacles bias the middle $\boldsymbol{v}_{gd}$ to generate θ_f (and thus $\boldsymbol{u}_{des}$) that move the arm away from obstacles. For instance, $\boldsymbol{v}_{gd}$ is biased between the two can obstacles (top right of Figure 5-26). Compared to the $\boldsymbol{v}_{gd}$ in the no obstacle case (top left of Figure 5-26), the selected $\boldsymbol{v}_{gd}$ rotate the arm CW to approximately position it through the middle point between the can centers to avoid collision. For the plane obstacles (bottom right of Figure 5-26), the selected $\boldsymbol{v}_{gd}$ rotate the arm CCW to be more parallel to the planes than perpendicular to them, relative to the no obstacle case (bottom left). Both sets of plots in Figure 5-26 show that the selected $\boldsymbol{v}_{gd}$ change with the incorporation of obstacles.

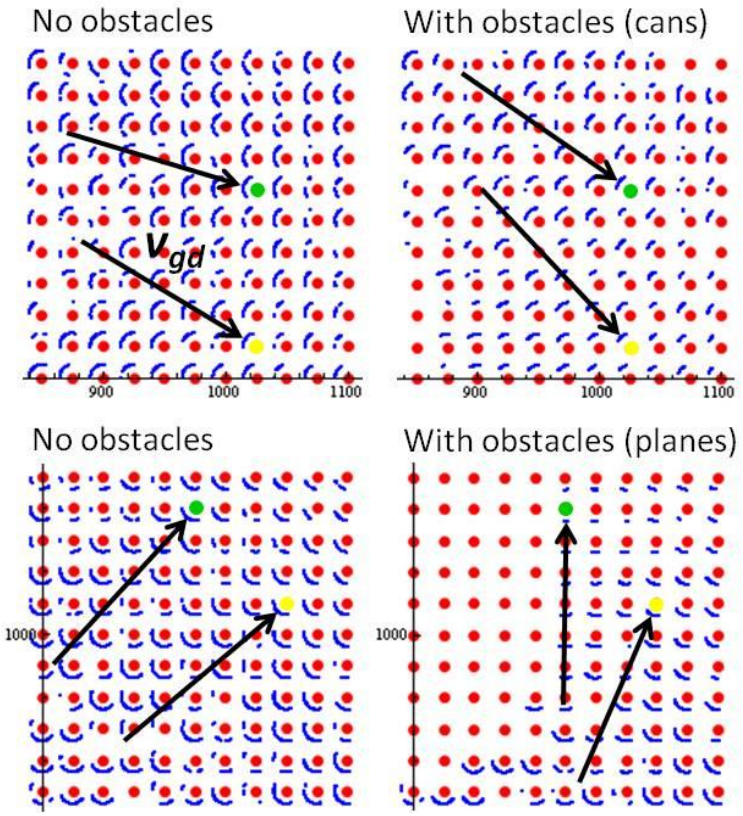


Figure 5-26: Direction plots for two areas between obstacles. The top row shows an area before and after the cans are added, while the bottom row shows an area before and after the planes are added. The green and yellow circles show the same grid points for each row of plots. There is a notable change in the selected grasping direction $\boldsymbol{v}_{gd}$ when obstacles are present.

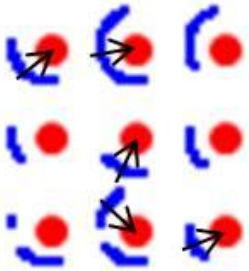


Figure 5-27: An inconsistent region of a directional plot. There are large differences between the adjacent $\boldsymbol{v}_{gd}$ selected along the path from the top corner to the bottom corner.

What cannot be seen from these snapshots is how the arm sometimes swings back and forth at times due to seemingly unnecessary internal joint motion. This occurs in

areas of the task plane that have inconsistent directional information, i.e. there is variation in the valid $\boldsymbol{v}_{gd}$ for adjacent points (Figure 5-27). Motions performed in the more continuous/consistent portions of the task plane do not usually exhibit this behavior. Furthermore, movements near obstacles have more constraints (i.e. biased grasping directions) as movements away from obstacles are freer to make larger movements. Other techniques for choosing a valid $\boldsymbol{v}_{gd}$ at a point may reduce this swinging motion and are discussed in Chapter 6, even though these motions may actually help maintain higher real-time AMI values. Maintaining higher manipulability may help ensure that the robot can complete tasks even as new or changing data about the task or environment arrives. For example, if a new obstacle is detected when already executing a motion task, a high AMI may better ensure that the robot can continue to complete the task.

5.3.2.2 Planning Top Grasp Trajectories

We can also demonstrate the capability to perform top grasp trajectories. This is done using the task planes for top grasps. Again, for these task planes, the percentage of valid top grasps tried is often either 0 or 100%. Furthermore, with the type of hands considered, the fingers of the hand/gripper are symmetric about 180º. For top grasp motion planning, therefore, the selection of θ_f for $\boldsymbol{u}_{des}$ is not very important since most θ_f are "good" if any top grasp is possible and a change in θ_f does not greatly affect the joint configuration.

For this motion planning, we again use the A* algorithm to plan a grid point path based on the manipulability index. Instead of using a method to select the pitch angle for each $\boldsymbol{u}_{des}$ at a grid point, we set θ_f to 0º. If not set in this way, a joint speed limit was often exceeded if there was a great difference between the θ_f chosen for adjacent grid points.

Figure 5-28 shows the task plane for the Motoman arm mounted at the center of the glovebox ceiling. This plane is a smaller region of the full workspace. The base frame is at a height of 943 mm and the task plane height is 101 mm. There are six obstacles in the workspace (see Table 5-6).

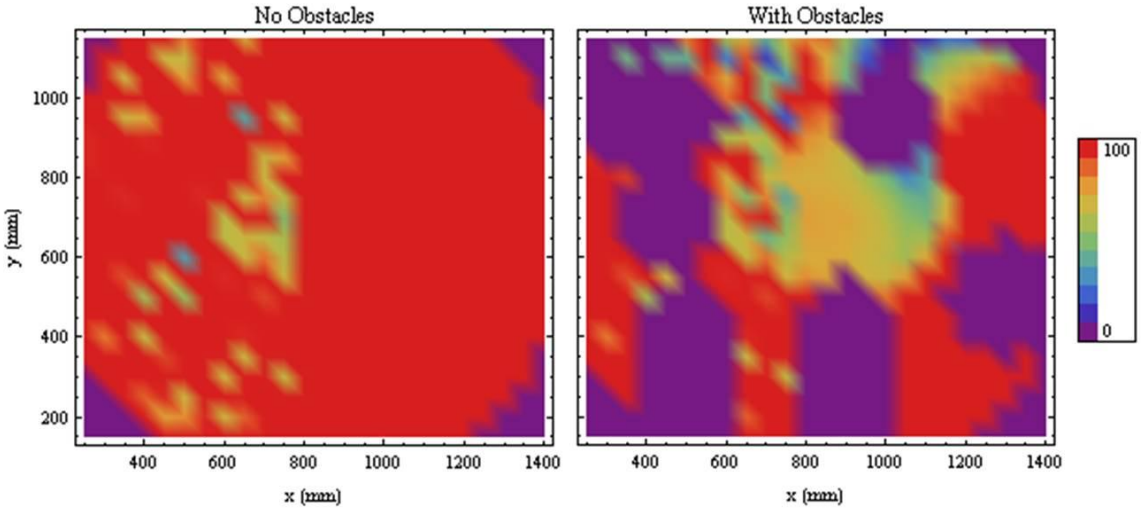


Figure 5-28: The task planes used for top grasp motion planning with the Motoman arm mounted from the center of the glovebox ceiling. On the right, six objects are added to the environment.

Table 5-6: The dimensions and locations of the six obstacles in the environment

Object	Radius (mm)	Height (mm)	Position of center (mm)		
			X	Y	Z
Can	52.4	136.5	1300	500	68.3
Can	20	136.5	450	700	68.3
Can	20	1143	1000	950	571.5

Object	Length (mm)	Height (mm)	Thickness (mm)	Position of center (mm)		
				X	Y	Z
Plane	300	250	1	500	300	125
Plane	300	250	1	900	300	125
Plane	200	500	1	400	950	250

Although we lack the directional plots generated for side grasp path planning, we can still obtain useful information from the task planes. Looking at Figure 5-28, the presence of obstacles greatly reduces the number of valid grasp configurations in the region directly below the base pose. The areas of greater manipulability are located outside this region. There is also a region of high manipulability on the far left of the plane that could be accessed if the small can between the two planes is removed. The combination of the top left plane and pole produces a region of varied manipulability between them. Finally, the top right corner shows the gradual decrease in manipulability as the hand is moved to the backside of the pole until an area of zero manipulability is reached.

For the paths generated in Figure 5-29, the AMI for the path with no obstacles is only about 10% higher than the AMI for the path with obstacles (see Table 5-7). With no obstacles, the AMI is high because the AMI values across the plane are also high. The path with obstacles, however, is over 40% higher than the AMI in the plane. This path is also physically valid while the no obstacles path reaches a joint limit in its motion. Switching the starting and end points finds slightly different paths for both but with the same AMIs as before. This time, the path with obstacles collides with the environment. This collision occurred because the IK found an initial configuration that collided with the pole obstacle. Had it found a different configuration, the path may have been physically valid since the grid point sequence was nearly the opposite of the original planned path.

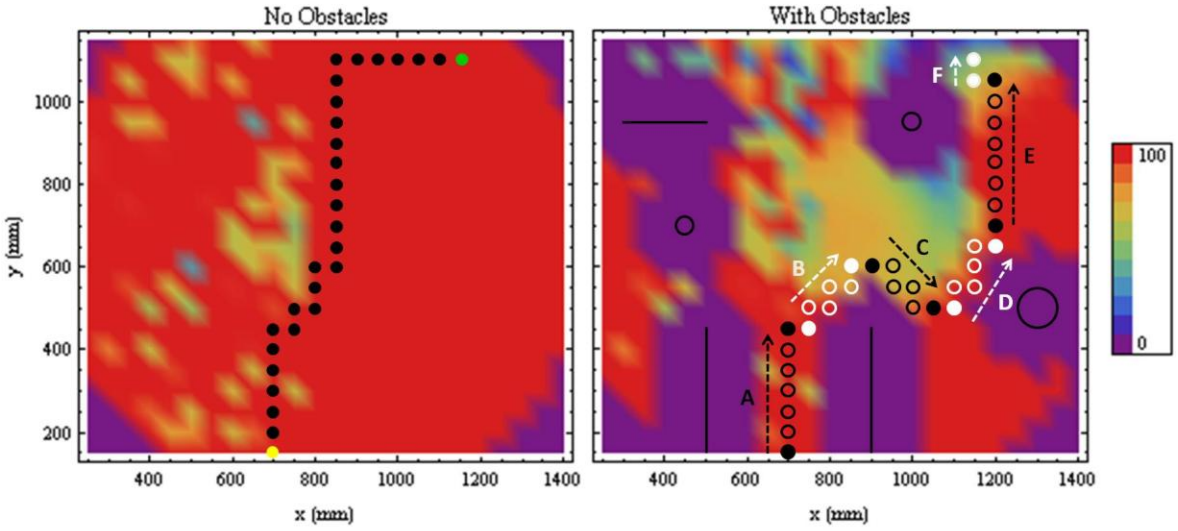


Figure 5-29: The path found in each plane using the A* algorithm. The starting grid point is in yellow and the final grid point is in green. The right task plane shows the gross movements A – F that are visualized in the Figure 5-30 images.

Table 5-7: AMI data for top grasp path planning

Path Test	Start Point		End Point		# Spheres Inside	AMI (%)	Plane AMI (%)	Valid Path?
	X	Y	X	Y				
No Obstacles	20	14	1	5	29	100	94.4	No (limits)
With Obstacles	20	14	1	5	35	90.8	49.5	Yes
No Obstacles	1	5	20	14	29	100	94.4	Yes
With Obstacles	1	5	20	14	35	90.8	49.5	No (collision)

Figure 5-30 shows the execution of the path with obstacles in Figure 5-29. The robot moves an object from one side of the glovebox to the other. The end location is behind a pole (relative to the starting position) connected to both the glovebox floor and ceiling. The arm first moves between two planes where the AMI is approximately 100% (Figure 5-30A). At the end of the planes is a region of decreased manipulability that is directly below the robot's base. Instead of moving directly through this region, the A* algorithm finds a path that goes around it to the right. To do this, the hand must make a

number of small maneuvers to get around the right plane (Figure 5-30B and C). The hand then moves toward the can on the right then around it (Figure 5-30D). At this point, the hand is far enough to the right to attain an approach trajectory that allows it to reach behind the pole (the final position) without colliding. The hand now moves through an area of high manipulability (Figure 5-30E) and approaches the backside of the pole. Once past the pole, the hand moves slightly to the left to reach the goal position (Figure 5-30F).

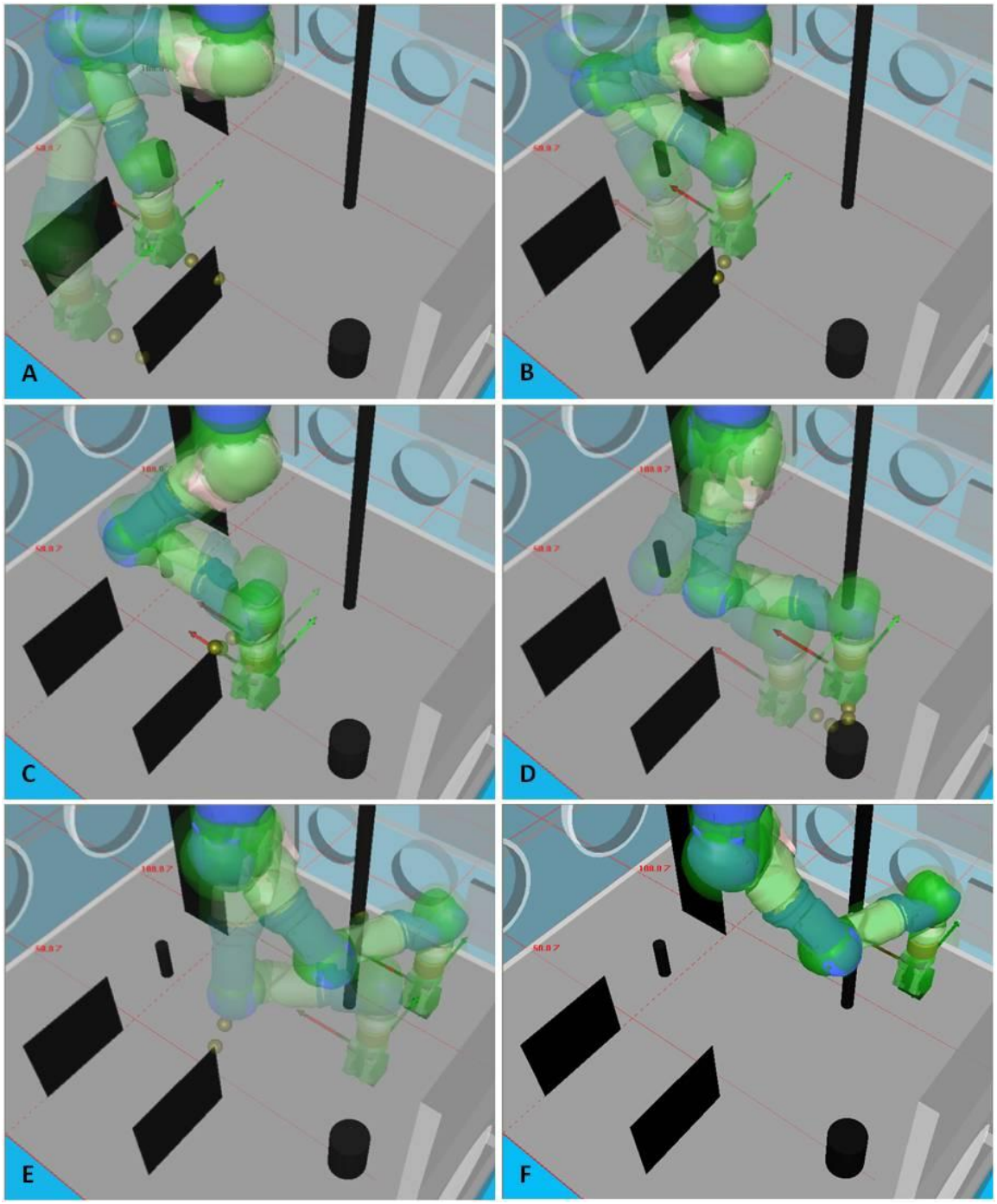


Figure 5-30: Snapshots of the top motion plan for the Motoman arm.

5.3.2.3 Motion Planning Issues and Evaluation

Although we have demonstrated a means of motion planning with task planes, there were some issues that occurred. The A* algorithm will always find a path as long as it does not start or end on a grid point of zero manipulability and there is not a region of zero manipulability that cannot be circumvented. If a path is found, however, that does not mean that it can be physically traversed if joint position limits, joint speed violations, and/or collisions occur. The root problem is differences in the $\boldsymbol{\varphi_i}$ found by the inverse kinematics during task plane generation and the $\boldsymbol{\varphi_i}$ used by the motion planner. When the IK uses the Jacobian to take the inverse, the Jacobian is a function of the previous kinematic joint state $\boldsymbol{\varphi_{pre}}$. Task planes are repeatable for the same IK because they calculate the inverse for the same sequence of $\boldsymbol{u}_{des}$. When the initial $\boldsymbol{\varphi_i}$ is computed for the initial $\boldsymbol{u}_{des}$ of the motion plan, $\boldsymbol{\varphi_{pre}}$ is most likely different than the previous state when the inverse of that same $\boldsymbol{u}_{des}$ was calculated in the task plane. Collisions in the configuration from the initial $\boldsymbol{u}_{des}$ sometimes occur because the collision-free $\boldsymbol{\varphi_i}$ found by the task plane is not the same one found by the motion planner.

Furthermore, after the motion planner finds an initial $\boldsymbol{\varphi_i}$, the subsequent configurations depend on this initial $\boldsymbol{\varphi_i}$. The difference in joint displacement of consecutive configurations is not very large. When a task plane is generated, however, configurations of adjacent grid points may have joint displacements that differ by a large amount. Thus, as the motion planner moves from grid point to grid point, it does not necessarily take on the valid $\boldsymbol{\varphi_i}$ found by the task plane. This means the motion planner could drive the joints to a limit – although there is a valid configuration at a point, the planned configurations do not go through this configuration. Joint speed violations were also noted, but these were due to the θ_f selection method of $\boldsymbol{u}_{des}$. The difference between

the selected θ_f of adjacent points may be large, requiring a rapid arm reconfiguration over the small distance between the grid points.

In general, if the manipulator begins its motion in a good configuration and a valid path is found in the search, then the actual trajectory has a high probability of success. This is reasonable because during operations, motions are only performed if the manipulator is operable (i.e. in a valid configuration.) Techniques for improving the success of planned motions are discussed in the next chapter.

5.4 IMPLEMENTATION SUMMARY AND CONCLUSIONS

In our Modified Task Plane Approach (MTPA), we employed the Capability Map Approach (CMA) of Zacharias [2007] with several simplifications, extensions and operational assumptions necessary for our application/task space. Through the implementation of our approach, we succeeded in demonstrating a wide variety of task plane applications. We developed procedures for system configuration management and optimization that did not rely on trial-and-error and used task planes as a tool for manipulator evaluation. We also developed a motion planning approach different than the method of Zacharias [2008]. Our method captured the influence of obstacles, supports non-linear trajectories, and utilizes an unconstrained Euler angle in the EEF orientation to select configurations that move the arm away from obstacles.

We still need to reconcile the source of task plane regions that show large variations. In these regions, the sectors of valid $\boldsymbol{v}_{gd}$ are very inconsistent between adjacent grid points. In general, the transitions between areas of high and low manipulability and the largest regions of manipulability are not as consistent as we would desire. This is likely due to our use of redundant manipulators and the IK algorithms utilized. Since multiple $\boldsymbol{\varphi}_i$ are possible for each $\boldsymbol{u}_{des}$, sometimes the IK finds a valid $\boldsymbol{\varphi}_i$

and other times the $\boldsymbol{\varphi}_i$ it chooses collides with an obstacle or exceeds joint limits, even for small changes in $\boldsymbol{u}_{des}$. One solution is better exploitation of manipulator redundancy. Instead of generating a single configuration for each TCP (which may or may not be valid), we search for additional configurations if the initial $\boldsymbol{\varphi}_i$ found is invalid. Inconsistencies are further discussed in Section 6.1.4.

The application of task planes for real-time use is still limited. Generating planes for the entire glovebox workspace took about 60 – 75 minutes. Planes that covered partial areas of the workspace varied from <10 to 45 minutes. However, if the planes are precompiled, trajectories can be quickly generated (less than 1 minute) and further investigation is necessary to ascertain the actual duration. Thus, the use of task planes is straightforward for static environments. In the case of dynamic environments, task planes need to be updated in areas where changes have occurred. If there is not an easy way of determining where updates are needed, the entire task plane must be re-generated. Working to improve real-time application is a topic of future research.

Zacharias [2009a] has also continued to build upon their previous work [2006, 2007, and 2008], incorporating capability maps with a grasp planner for online grasp planning. Essentially, the grasp planner uses the capability map to determine a grasp's reachability by a robotic arm. In this approach, obstacles are incorporated into the map formulation by establishing an obstacle's *region of influence* with respect to collisions. This region of influence is then subtracted from the original task plane.

The incorporation of obstacles coincides with our map development. We also addressed more confined and cluttered environments and considered collisions with the environment, while Zacharias did not. However, Zacharias better exploited manipulator redundancy to find valid grasping configurations that did not collide. This should lead to more regular workspace structure (less variation between adjacent points). We only

generated a single configuration for each TCP, and the lack of regularity sometimes led to unnecessary movements while executing trajectories.

Additionally, when the region of influence is subtracted from the original capability map, the resulting map is not a perfect representation of reachability, so configurations still need to be checked for collisions. [Zacharias, 2009a] We also needed to confirm whether generated trajectories were actually collision-free. Subtracting obstacle influence also damages the formerly present regular structure in the map for valid grasps. This leads to higher shape errors for volumes containing valid grasps at a point which means more unreachable grasps may be classified as reachable. In our obstacle incorporation, we do not worry about this type of error and also require no additional steps for obstacle inclusion – they are built right into the original map. Zacharias currently computes obstacle influence offline, but our obstacles are included when the map is generated and thus already integrated into any method we develop to improve real-time capability. Overall, both methods represent significant progress in extending capability map functionality and working toward the goal of an online method for working in dynamic environments.

Zacharias [2009a] has a large impact on the development of a grasp planner, but they also made improvements for using capability maps in motion planning, including extensions to 3-D. [Zacharias, 2009b] Both of these topics will be further described in Chapter 6. Due to the developments beyond Zacharias [2006, 2007 and 2008], our future work with capability maps will build upon the work of Zacharias [2009].

Chapter 6 **Future Work**

Our work builds on that of Zacharias [2006, 2007, and 2008]. Although we made a number of extensions to that work, there are many areas where further investigation would be beneficial. A logical step would be to apply our methods to 3-D space since we reduced the 3-D approach proposed by Zacharias [2007] to a 2-D method for this research. As we discuss modifications and extensions to our current 2-D task plane implementation and motion planning capability, we will mention how to extend current and proposed functionality to 3-D space where applicable.

6.1 TASK PLANE IMPROVEMENTS AND EXTENSIONS

There are a few variations and extensions that could help task planes more accurately indicate a manipulator's capabilities, creating a more robust grasping strategy. Many of these additions increase the computational time for task plane generation, making it less feasible for real-time use.

6.1.1 Complex Grippers and Objects

In our approach, a significant manipulation assumption is the use of simple grippers and objects with symmetries. Additionally, we did not incorporate a grasp analysis but assumed that adequate grasps were attained by closing the grippers or fingers around the object of interest. The only planning necessary was positioning the EEF around the object (i.e. locating the tool point on the object's center). In practice, we need to generate the task plane with the object to be grasped as an obstacle. Then we could

plan the motions necessary to move the hand around the object without collision, closing the fingers/gripper when the hand is properly positioned for the grasp.

In general, utilizing a complex hand would boost object manipulability at a point in space due to the increased system DOF. For objects with more complex geometries, grasp constraints could exclude certain grasping configurations. However, the net effect would be positive since a complex hand would add valid grasping configurations (and thus manipulation) that was not possible before. The meshing of manipulator DOF and EEF DOF to achieve optimal task configurations is an interesting topic for future work.

Using complex grippers and objects requires trial TCPs other than those for side and top grasps. One option is to use the full reachability sphere technique employed by Zacharias [2007]. This does not require an extensive amount of programming time, but does add significantly to the computational time of a task plane. Depending on the object geometry, we may need to generate a few parallel task planes that intersect the object in order to determine the best plane to make the grasp. Basically, we need to extend functionality to include a larger set of valid grasping configurations. Using our object and gripper simplifications, we found a small subset of the set of all valid tool frame locations for gripping. We need to generalize our approach to include the set containing the full space of valid tool frames for an object/grasper pair. In operation, this set is generated by a separate component and then fed to a capability map component that determines if a manipulator grasping configuration is possible for each grasp solution in the set. An algorithm that simplifies the grasp space, for example, from a sphere to a smaller subset such as a plane or surface would help reduce the computation time required with the components implemented in this effort.

The work of Zacharias [2009a] described in Section 5.4 has already demonstrated one way of using capability maps with a grasp planner. This approach had many

advantages. When a capability map is used, unreachable grasps can be detected and discarded early on. Collision avoidance is performed without actually including the robot and object models into the grasp planner and the planner is also no longer tightly coupled to the arm's inverse kinematics. Thus, to adapt the grasp planner to a new robot, only its capability map is needed. Similarly, we need to investigate how our task plane implementation can be incorporated into a full grasping strategy.

6.1.2 Collision Model

In our manipulator collision model, we used cylispheres. The size of these cylispheres is often chosen to envelope the entire link of interest, leading to a very conservative approximation for some link geometries. At times, this eliminated grid points and configurations that would not have caused a collision with better modeling. These eliminations lead to a small inaccuracy in the actual manipulability in a region and make motion planning in the presence of obstacles slightly harder. As mentioned in Chapter 3, more accurate modeling techniques exist, such as using STLs and the SOLID libraries to perform collision detection. These models give a more accurate description of the distance between a manipulator and its environment leading to greater ability to utilize manipulability in the workspace.

6.1.3 Motion Planning Check

The task plane gives information about valid grasping configurations at a reachable point assuming that a valid path can be found from some current configuration to any valid configuration at that point. A better motion planning check could be implemented for each $\boldsymbol{u}_{des}$ generated at a grid point to make this a more robust

assumption. We implemented a joint interpolated 5th order polynomial check from a reference configuration $\boldsymbol{\varphi}_{ref}$ to each $\boldsymbol{\varphi}_i$.

A Cartesian interpolated motion check would be more appropriate for the type of trajectories generated. We also need to incorporate a joint limit check on the trajectory, something the OSCAR motion planner does not currently perform. Thus, given $\boldsymbol{u}_{ref}$, we can check for the existence of a collision-free path to $\boldsymbol{u}_{des}$ that also avoids joint limits. In theory, the motion could be from any valid configuration in the plane to $\boldsymbol{u}_{des}$. To get around this, we can instead check that motion is possible in some small area about $\boldsymbol{u}_{des}$ or randomly sample the workspace for several $\boldsymbol{u}_{ref}$, to perform the check for. If the number of valid paths found passes some criterion, then $\boldsymbol{u}_{ref}$ is valid.

6.1.4 Manipulability Index

The manipulability index could also incorporate other criteria to give a more robust indication of manipulability by, for example, checking the Jacobian condition number or JRA criterion to detect proximity to singularities or joint limits, respectively. Favorable criteria values would increase the manipulator's capabilities and the manipulability index.

Most of our generated task planes contained areas where the manipulability index varied greatly from point to point. Large areas of uniform manipulability were hard to find and the transition between low and high manipulability often had local peaks and valleys. To fix this, we need a better understanding of what causes the non-uniformity. For example, the closed-form IK for the LWA3 arm generated a very smooth task plane while the JRA IK task plane had more variation. This may be due to the analytical solution method of the former and the numerical solution method of the latter. If the

appearance is an artifact of our grid resolution, then we need to decrease the grid spacing or derive a method that better approximates the manipulability between grid points.

6.1.5 Target Location

There are a few ways the target location method can be improved. Currently, the location method only considers the manipulability index at the grid points. However, the workspace region with the highest manipulability may be difficult to access or frequent motions may be necessary between two targets, so the area between them should have as high an AMI as possible. For the cabinet opening problem, we developed a single target that contained targets for objects and the likely area of trajectories, but this incorporates no information about the actual trajectories that may be used. Incorporating a better motion plan check in the generation of the task plane such as those just proposed could be a solution. We could also couple the task plane motion planning capability with the best location search. Using the A* algorithm to plan a number of paths to $\boldsymbol{u}_{des}$, we calculate the AMIs given by these paths and combine them with the AMI of the targets. Support for 3-D trajectories would add this effort and is discussed later.

With a group of targets, all grid point indexes are weighed equally regardless of the shape they lie inside. If some targets are used more frequently or require more dexterous manipulation, we can give larger weights to indices that are within these targets, calculating a weighted AMI. We can apply the same idea to selecting the best location of targets independently. We only dealt with groups of targets that were fixed with respect to one another. With our current algorithm, searching for each target separately would likely result in overlapping locations in the workspace. If the targets are searched for in order of importance, we then restrict subsequent searches from overlapping with a region containing the grid points related to previous best locations

found. Additionally, we can handle the relocation of targets after the incorporation of obstacles using a similar technique that restricts any target from overlapping obstacle regions.

6.2 MOTION PLANNING IMPROVEMENTS AND EXTENSIONS

We successfully implemented a skeleton/graph approach to motion planning for task planes. Our main goal was to demonstrate that this capability exists, so there are many areas to investigate to make this capability more robust.

6.2.1 Complex Motions

Although we extend the work of Zacharias [2008] to include non-linear trajectories without fixed EEF orientations, non-planar motions should also be implemented. Less constrained motions lead to more trajectory options and improve the maneuvering of the arm around obstacles. Manipulability is increased and the system can now complete more tasks. Furthermore, our current capability accomplishes gross motion planning. At the initial $\boldsymbol{u}_{des}$ in the path plan, we assumed the grasp could be executed to put the hand at $\boldsymbol{u}_{des}$. In practice, depending on the object, we also need local motion planning to make the grasp. Gross planning can be used to move the EEF to an approach configuration $\boldsymbol{u}_{ap}$ from which local motion planning moves the EEF to $\boldsymbol{u}_{des}$. If the task plane is generated to handle complex hands and geometries (Section 6.1.1), then the task plane should provide enough data to execute this motion.

Developing 3-D trajectories based on the MTPA could involve the planning of motions that span adjacent horizontal task planes. Using the graph search approach, we could implement an algorithm that traverses a 3-D grid of points. In our previous

motions, the EEF orientation only had one free variable θ_f, as the other variables defined either a side or top grasp. We can relieve these constraints on the EEF depending on the object type or when moving without an object. This requires a new method of selecting which EEF orientation to attain, as valid $\boldsymbol{v}_{gd}$ no longer lie in sectors and TCPs with rotations about valid $\boldsymbol{v}_{gd}$ are also possible. For our implementation, improved 2-D and 3-D $\boldsymbol{v}_{gd}$/TCP selection methods will be discussed in the next section.

Zacharias [2009b] completed another approach for selecting TCPs and generating 3-D trajectories using capability maps. Essentially, the trajectory is superimposed on the workspace discretization underlying the reachability sphere maps. The trajectory frames are then mapped to frames on the reachability spheres to generate a trajectory pattern. The pattern is then moved across the capability map and the correlation between the pattern and the valid grasping directions on the spheres the trajectory passes through/by is determined. The correlation results are used to determine where the trajectory should be performed in the capability map.

6.2.2 Grasping Configuration Selection Method

To demonstrate the motion capability of task planes, we chose a simple technique for selecting grasping configurations. Due to our simplifications, this reduced to selecting θ_f to define each $\boldsymbol{u}_{des}$. Given the valid $\boldsymbol{\varphi}_i$ at a point from each θ_f, we could select θ_f by using a combination of criteria to rank each corresponding $\boldsymbol{\varphi}_i$. However, in this effort, we wanted to demonstrate the motion planning capability solely using information from the task planes. Without incorporating criteria, we can develop other techniques of choosing θ_f based on task plane data. We will first discuss other methods for selecting θ_f and then address techniques for choosing configurations after many simplifications are removed.

The selection could be aided by a heuristic. Zacharias [2006] found that a heuristic could locate regions of valid grasping directions also identified using a brute force method. In the directional plots without obstacles, we saw that the valid $\boldsymbol{v}_{gd}$ were influenced by the location of the grid point relative to the first pitch joint. Drawing a vector $\boldsymbol{v}_{pj}$ from the first pitch joint to the grid point of interest either intersects the longest continuum of valid $\boldsymbol{v}_{gd}$, intersects one of multiple sectors or contains a valid sector to its left and right. The arm usually swings back and forth during motion between adjacent points if one contains a single sector of $\boldsymbol{v}_{gd}$ and the other contains multiple sectors. If we always select a $\boldsymbol{v}_{gd}$ on the right side of $\boldsymbol{v}_{pj}$ at a grid point, then we decrease the swinging of the arm. This is similar to grasping in the area *G* in Zacharias' heuristic (Figure 3-20). If no valid $\boldsymbol{v}_{gd}$ are in that location, then we can revert back to our original technique.

In environments with obstacles, the valid $\boldsymbol{v}_{gd}$ are biased according to obstacle locations, so the heuristic may not be as effective. Instead, if we encounter multiple sectors at the next gird point, we choose the sector that contains a $\boldsymbol{\varphi}_i$ most similar to the current $\boldsymbol{\varphi}_i$. Another technique could select the $\boldsymbol{v}_{gd}$ at the center of the overlap of adjacent grid point sectors. Since the sector information is likely to reflect the arm's grasping preferences, which would guide the development of a heuristic, the best solution probably lies in using sector information at grid points.

For task planes generated with unconstrained EEF orientations, the selection method is more complex. To correlate with our 2-D approach, in 3-D we could select the $\boldsymbol{v}_{gd}$ in the center of the volume containing all valid $\boldsymbol{v}_{gd}$. These volumes would be similar to the cone and cylinder shapes selected by Zacharias [2007] and seen in Figure 3-7. Many of the same issues from the 2-D case occur in 3-D. For instance, there could be two volumes where the valid $\boldsymbol{v}_{gd}$ are clustered. Similar selection approaches could be used: incorporate other criteria to rank $\boldsymbol{\varphi}_i$, choose $\boldsymbol{v}_{gd}$ that keep joint displacements at a

minimum between grid points, etc. There is also the correlation technique proposed by Zacharias [2009b].

6.2.3 Path Search

There are a few changes to the A* search algorithm that could improve the quality of the paths found. Currently, the cost function only factors in the manipulability index M at each point to determine the path. Adding other criteria (JRA, MOT, etc) to the cost function could improve the path found. This is similar to using other criteria with the AMI when making design decisions. The values for these criteria would be derived from the valid $\boldsymbol{\varphi}_i$ at that location. The cost could also incorporate data from adjacent grid points, looking for overlap in valid $\boldsymbol{v}_{gd}$. Overlap in $\boldsymbol{v}_{gd}$ of adjacent points could mean small changes between the $\boldsymbol{\varphi}_i$ generated by those directions at each point. Some of these options require the coupling of the grasping configuration selection method and the cost function.

Other search algorithms like the depth-first search (Section 3.3.4) could be implemented. This search is governed by a predetermined, preferred search direction from the current configuration. Although, the path may not be the shortest with respect to cost, it could bias the path direction to avoid trouble areas.

Finally, the search algorithm could be utilized to conduct a form of dynamic motion planning that could be useful for real-time implementation. If a workspace grid is established but no task plane/reachability spheres have been generated, a path can still be found by finding the manipulability index at each point as needed. The A* algorithm does not visit every grid point since a heuristic is used to guide it toward the final location. This process would adapt to changes in environment between each path plan

and has a lower computational time than generating a new task plane. The computational time would need to be analyzed for actual real-time use.

6.2.4 Motion Success

Trajectories developed from our motion planning method are not always physically attainable. This is the case with most motion planners and developing techniques to improve the number of successful trajectories is worthwhile. We also need to evaluate the success of our motion planning by calculating the paths found out of the total number of valid paths and the generated paths that are physically valid. This provides a means to compare our motion planning approach to others.

One of the best solutions to improving success could be incorporating a more robust configuration selection method (Section 6.2.2). Changes made to the generation of task planes (Section 6.1) or the formulation of the path search (Section 6.2.3) could also help. We could also further modify the path search to store the best path and a few other good paths. With more paths sampled, the probability of a valid path increases.

A final solution is providing more task plane data to the motion planner or creating our own trajectory generator. Particularly, the joint solutions $\boldsymbol{\varphi}_i$ calculated from valid $\boldsymbol{u}_{des}$ during plane generation is valuable, although it takes a large amount of storage space. These $\boldsymbol{\varphi}_i$ are guaranteed to be physically valid if used in a trajectory. However, the motion planner may not use these $\boldsymbol{\varphi}_i$ when it applies inverse kinematics to each $\boldsymbol{u}_{des}$ since the manipulator is redundant. If the valid $\boldsymbol{\varphi}_i$ are saved, then we can demand that the trajectory goes through them. There is still the possibility that large joint displacements may occur between valid $\boldsymbol{\varphi}_i$ at adjacent points leading to movements that span large regions on the workspace. Since the valid configurations between $\boldsymbol{u}_{des}$ are not precisely known, this may lead to collisions or other unwanted conditions. The task plane itself

could be biased to choose valid configurations at a point that are a small perturbation of adjacent solutions or the search algorithm could change to factor overlap of valid $\boldsymbol{v}_{gd}$ in adjacent points (Section 6.2.3).

6.2.5 Real-Time Summary

Some of the proposed future work discussed techniques to make our algorithm better suited for real-time application. Below is a summary of a few possible solutions:

- More efficient collision check
- Dynamic path find that generates reachability spheres only if traversed by the A* algorithm
- Re-generate workspace areas only where obstacles have moved from/to
- Simplification of valid grasp space

The effect on real-time application is a major consideration when determining the research direction within future work topics.

6.3 CONCLUSION

Using task planes, we can successfully choose "good" manipulator configurations based on manipulability for grasping before and after executing a motion trajectory. This is particularly useful for decision making concerning grasping configurations for dynamic objects. This development helps us make a small incremental step in system autonomy. In regard to path planning, additional work is necessary before a more robust step is made in the decision making process. Most areas of future work are related to motion planning

and its connection to a grasping strategy for the manipulator. Some improvements require changes to several aspects of the entire method: task plane generation, path search algorithm, and grasping configuration selection. Overall, information about manipulability in the workspace is an insightful tool for developing grasping strategies for changing tasks.

Appendix

Below are the header files for the `ReachSphere` and `Target` classes.

File: `ReachSphere.h`

```
#include "Base/Base.h"
#include "Math/Vector.h"
#include "Math/Xform.h"
#include "ForwardKinematics/FKJacobian.h"
#include "InverseKinematics/IKJReconfig.h"
#include "InverseKinematics/IKJAvoidLimits.h"
#include "InverseKinematics/PCLW3ClosedFormIK.h"
#include "ObstacleAvoidance/WorkCell.hpp"
#include "C:/OSCAR/MotionPlanning/PathPlanning/PathPlanner.h"
#include "C:/OSCAR/MotionPlanning/MotionPlanner.h"


namespace OSCAR {

	class ReachSphere : public Base {

	public:

	ReachSphere();
	ReachSphere(int num, double x, double y, double z);
	double BuildSphere(IKJReconfig<> ikj, FKJacobian fkj, WorkCell*
		workcell, Matrix limits, double angles[]);
	Vector BuildSphere(IKJReconfig<> ikj, FKJacobian fkj, WorkCell*
		workcell, Matrix limits);
	double BuildSphere(PCLW3ClosedFormIK ikj, FKJacobian fkj,
		WorkCell* workcell, Matrix limits, double angles[]);
	bool BuildSphere(IKJReconfig<> ikj, FKJacobian fkj, WorkCell*
		workcell, unsigned int dof, Matrix limits, double
		angles[]); //Old version not used
	double BuildSphere(IKJReconfig<> ikj, FKJacobian fkj, WorkCell*
		workcell, MotionPlanner planner, Matrix limits, double
		angles[]); //Old version not used
	void SetAngle(int index, double angle);
	double GetAngle(int index);
	double CalcIndex();
	int CompareAngles(ReachSphere sphere);
	Vector FindSector(Matrix limits);
	void operator=(ReachSphere rhs);
	void SetX(double x);
	void SetY(double y);
	void SetZ(double z);
```

```
        void SetManIndex(double index);
        double GetX();
        double GetY();
        double GetZ();
        double GetIndex();
        int GetTag();
        const int GetIter();

        protected:

        double X;
        double Y;
        double Z;
        static const int Iter = 72;
        int Tag;
        double Angles[Iter];
        double ManIndex;

        };

}
```

File: `Target.h`

```
#include "Base/Base.h"
#include "ReachSphere.h"
#include "Math/Vector3.h"

namespace OSCAR {

        class Target : public Base {

        enum TShape {RECT, CIRCLE};

        public:

        Target();
        Target(double x, double y, double z, double length, double width,
              double theta);
        Target(double x, double y, double R);
        bool Inside(ReachSphere s);
        Target MoveToGridPoint(ReachSphere s, double phi);
        void operator=(Target rhs);
        void SetX(double x);
        void SetY(double y);
        void SetZ(double z);
        void SetLength(double length);
        void SetWidth(double width);
        void SetTheta(double theta);
        double GetX();
        double GetY();
```

```
        double GetZ();
        double GetLength();
        double GetWidth();
        double GetTheta();
        TShape GetShape();
        Vector3 GetP1();
        Vector3 GetP2();
        Vector3 GetP3();
        Vector3 GetP4();

        protected:

        double X;
        double Y;
        double Z;
        Vector3 p1;
        Vector3 p2;
        Vector3 p3;
        Vector3 p4;
        double TLength; //Along x-direction
        double TWidth; //Along y-direction
        double Theta; //Between global x-axis and length, in RADIANS
        TShape shape;

        };

}
```

References

Abdel-Malek, K., Yu, W., and Yang, J., "Placement of robot manipulators to maximize dexterity," *International Journal of Robotics and Automation*, 19 (1), pp. 6 – 14, 2004.

Achint, A. and Tesar, D., "Task Based Global Motion Planning of Multiple Manipulators in Time-Varying Environments," *Master's Thesis*, The University of Texas at Austin, 2009.

Argonne National Laboratory, "ARRA-funded work at Alpha Gamma Hot Cell Facility," Web address: http://www.flickr.com/photos/argonne/3859726946/in/photostream/, August 2009.

Ben-Shahar, O. and Rivlin, E., "Practical Pushing Planning for Rearrangement Tasks," *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 4, pp. 549 – 565, August 1998.

Borenstein, J. and Koren Y., "The Vector Field Histogram – Fast Obstacle Avoidance for Mobile Robots," *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 3, pp. 278 – 288, 1991.

Cao, H., Ling, Z., Zhu, J., Wang, Y., and Wang, W., "Design Frame of a Leg Exoskeleton for Load-Carrying Augmentation," *Proceedings of the 2009 IEEE International Conference on Robotics and Biomimetics*, Guilin, China, December 2009.

Chiu, S., "Task Compatibility of Manipulator Postures," *International Journal of Robotics Research*, Vol. 7, No. 5, pp. 13 - 21, October 1988.

Chiu, S., "Control of Redundant Manipulators for Task Compatibility," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1718 – 1724, Raleigh, NC, March 1987.

Cox, D., Legault, J., Turner, C., and Tesar, D., "Automated Plutonium Processing Workcell Technology," *Proceedings of the American Nuclear Society*, 1999.

Craig, J.J., "Introduction to Robotics," Addison Wesley, 1989.

de M. Martins, A., Dias, A., and Alsina, P., "Comments on manipulability measure in redundant planar manipulators," *IEEE 3rd Latin American Robotics Symposium*, Santiago, Chile, October 2006.

Dollar, A.M., Jentoft, L.P, Gao, J.H., and Howe, R.D., "Contact sensing and grasping performance of compliant hands," *Autonomous Robot*, 28 (1), pp. 65 – 75, 2010.

Dollar, A.M. and Howe, R.D., "Towards grasping in unstructured environments: Grasper compliance and configuration optimization," *Advanced Robotics*, 19 (5), pp. 523 – 544, 2005.

Dubey, R. and Luh, J.Y.S., "Redundant Robot Control Using Task Based Performance Measures," *Journal of Robotic Systems*, Vol. 5, Issue 5, pp. 409 – 432, 1988.

Ferrari, C. and Canny, J., "Planning Optimal Grasps," *Proceedings of the International Conference on Robotics and Automation*, pp. 2290 – 2295, 1992.

Few, D. A., Bruemmer, D. J., and Walton, M. C., "Improved Human - Robot Teaming through Facilitated Initiative," *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (ROMAN)*, Hatfield, United Kingdom, September 2006.

Harden, T. and Turner, C. J., "Robotics for Nuclear Material Handling at LANL: Capabilities and Needs," *2009 ASME IDETC/CIE Conference*, San Diego, CA, August 2009, LA-UR-09-01157.

Harden, T. and Pittman, P., "Development of a Robotic System to Clean Out Spherical Dynamic Experiment Containment Vessels," *Proceedings of the ANS 12th International Topical Meeting on Robotics and Remote Systems*, March 2008.

Harden, T. and Tesar, D., "The Implementation of Artificial Potential Field Based Obstacle Avoidance for a Redundant Manipulator," *Master's Thesis*, The University of Texas at Austin, 1997.

Harmo, P., Taipalus, T., Knuuttila, J., Vallet, J., and Halme, A., "Needs and Solutions – Home Automation and Service Robots for the Elderly and Disabled," *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Alberta, Canada, August 2005.

Hester, R. and Tesar, D., "A Criteria-Based Approach to Grasp Synthesis," *Master's Thesis*, The University of Texas at Austin, 1998.

Heyes-Jones, J., "Justin Heyes-Jones web pages – A* Algorithm," Website: http://www.heyes-jones.com/astar.html, 2001-2005.

Hill, B.M. and Tesar, D., "Design of Mechanical Properties for Serial Manipulators," *Internal Report*, The University of Texas at Austin, August 1997.

Hulse, A., Kulkarni, A., O'Neil, B., Pryor, M., and Kapoor, C., "Improved Grasping Strategies for Flexible Manufacturing and Mobile Manipulation," *2009 American Nuclear Society Winter Meeting*, Washington, DC, November 15-19, 2009.

Hwang, Y.K. and Ahuja, N., "Gross Motion Planning: A Survey," *ACM Computing Surveys*, 24 (3), pp. 219 – 291, September 1992.

Ito, M, Kawatsu, K., and Shibata, M., "Maximal admission of joint range of motion based on redundancy resolution for kinematically redundant manipulators," *2010 SICE Annual Conference*, Taipei, Taiwan, August 2010.

Kapoor, C., Jung, E., Rabindran, D., and Pehl J., "Modular Small Automation for Glovebox Operations," Robotics Research Group Presentation, The University of Texas at Austin, 2002.

Kapoor, C. and Tesar, D., "A Reuseable Operational Software Architecture for Advanced Robotics," *Dissertation*, The University of Texas at Austin, 1996.

Kavraki, L.E., Svestka, P., Latombe, J.C., and Overmars, M.H., "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces," *IEEE Transactions on Robotics & Automation*, 12 (4), pp. 566 – 580, 1996.

Klein, C.A., "Use of Redundancy in the Design of Robotic Systems," *Robotics Research: The Second International Symposium*, Eds. H. Hanafusa and H. Inoue, Cambridge: MIT Press, pp. 207-214, August 1985.

Knoll, J. and Tesar, D., "Complete Workcell Modeling for Robot Control and Coordination," *Master's Thesis*, The University of Texas at Austin, 2007.

Kulkarni, A., Kapoor, C., Pryor, M., Kinoshita, R., and Bruemmer, D., "Software framework for mobile manipulation," *ANS 2nd International Joint Topical Meeting on Emergency Preparedness & Response and Robotic & Remote Systems,* Albuquerque, NM, March 2008.

LaValle, S.M., "Rapidly-Exploring Random Trees: A New Tool for Path Planning," *Technical Report 98-11*, Computer Science Department, Iowa State University, October 1998.

Lee, K. and Buss, M., "Redundancy resolution with multiple criteria," *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, October 2006.

LeGoullon, A. and Tesar, D., "Configuration Management of Robotic Workcells," *Master's Thesis*, The University of Texas at Austin, December 1997.

Lozano-Perez, T., Jones, J.L., Mazer, E, and O'Donnell, P.A., "Task-Level Planning of Pick-and-Place Robot Motions," *Computer*, Vol. 22, Is. 3, pg. 21-29, March 1989.

March, P. and Tesar, D., "Criteria Based Motion Planning," *Master's Thesis*, The University of Texas at Austin, 2004.

Noakes, M. W. and Haley, D. C., "Deployment of Remote Systems in U.S. Department of Energy Decontamination & Decommissioning Projects," *Spectrum 2000*, September 2000.

Nuclear Weapon Archive, "Plutonium Manufacturing and Fabrication," Web address: http://nuclearweaponarchive.org/Library/Plutonium/Glovebx.jpg

O'Neil, B., Schroeder, K., Williams, J., and Pryor, M., "A Modular Research Approach to Incremental Improvements in Manipulator Automation," *2010 ANS Student Conference*, Ann Arbor, Michigan, April, 2010.

Patel, R. and Shadpey, F., "Control of redundant robot manipulators," *Lecture Notes in Control and Information Sciences*, Vol. 316, 2005.

Perry, B. and Tesar, D., "The Development of Distance Functions and Their Higher-Order Properties for Artificial Potential Field-Based Obstacle Avoidance," *Master's Thesis*, The University of Texas at Austin, 1995.

Pholsiri, C, Kappor, C., and Tesar, D., "Manipulator task-based performance optimization," *Proceedings of ASME 2004 Design Engineering Technical Conference and Computers and Information in Engineering Conference*, Salt Lake City, UT, September 2004.

Pryor, M., "Transitional levels of autonomy for human-machine cooperative systems," *NSF Proposal*, 2010.

Pryor, M., Taylor, R., Kapoor, C., and Tesar, D., "Generalized software components for reconfiguring hyper-redundant manipulators," *IEEE/ASME Transactions on Mechatronics*, Vol. 7, No. 4, pp. 475 – 478, December 2002.

Pryor, M. and Tesar, D., "Complex Task Completion with Redundant Serial Manipulators," *Master's Thesis*, The University of Texas at Austin, 1999.

Rajan, R., "Foundation Studies for an Alternate Approach to Motion Planning of Dynamic Systems," *Master's Thesis*, The University of Texas at Austin, 2001.

Seward, W. D. and Bakari, M. J., "The Use of Robotics and Automation in Nuclear Decommissioning," *22nd International Symposium on Automation and Robotics in Construction*, Ferrara, Italy, 2005.

Shimoga, K., "Robot Grasp Synthesis Algorithms: A Survey," *International Journal of Robotics Research*, Vol. 15, No. 3, pp. 230 – 266, June 1996.

Spencer, A., Pryor, M., Kapoor, C., and Tesar, D., "Collision Avoidance Techniques for Tele-Operated and Autonomous Manipulators in Overlapping Workspaces," *2008 IEEE International Conference on Robotics and Automation*, Pasadena, CA, May 19-23, 2008.

Spencer, A. and Tesar, D., "Obstacle Avoidance for Multiple Robot Systems," *Master's Thesis*, The University of Texas at Austin, 2007.

Sreedhar, R. and Tesar, D., "Potential Function Based Obstacle Avoidance Algorithm for Manipulators with Extra Degrees of Freedom," *Report to the DOE under Grant No. DE-FG02-86NE37966*, The University of Texas at Austin, August 1990.

Swint, E. and Tesar, D., "Collision Detection and Obstacle Avoidance for Robotic Manipulators," *Master's Thesis*, The University of Texas at Austin, 2004.

Tesar, D. and Matthew, G., "The Dynamic Synthesis, Analysis, and Design of Modeled Cam Systems," Lexington Books, D. C. Heath & Company, 1976.

Tisius, M., Pryor, M., Kapoor, C., and Tesar, D., "An empirical approach to performance criteria for manipulation," *Journal of Mechanisms and Robots*, Vol. 1, No. 3, August 2009.

Tournassoud, P., Lozano-Perez, T., and Mazer, E., "Regrasping," *Proceedings of IEEE International Conference on Robotics and Automation*, Los Alamitos, CA, 1987.

van den Bergen, G., "User's Guide to the SOLID Collision Detection Library," http://dtecta.com/, Ver. 3.5, 2003.

Venables, M., "Ford's flexible future," *IEE Manufacturing Engineer*, Vol. 84, Is. 6, pp. 36 – 39, December 2005/January 2006.

Venables, M., "Small is beautiful," *IEE Review*, Vol. 51, Is. 3, pp. 26 – 27, March 2005.

Walsh, C. J., Pasch, K., and Herr, H., "An autonomous, underactuated exoskeleton for load-carrying augmentation," *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, October 2006.

Williams, J.M., "Glovebox Automation for MOX Fuel Fabrication," *2009 LANL Student Symposium*, LA-UR-09-04881, Los Alamos, NM, August 2009.

Xiong, C., Wang, M., Tang, Y., and Xiong, Y., "Compliant grasping with passive forces," *Journal of Robotic Systems*, 22 (5), pp. 271 – 285, 2005.

Yoshikawa, T., "Manipulability of robotic mechanisms," *Robotics Research: The Second International Symposium*, Eds. H. Hanafusa and H. Inoue, Cambridge: MIT Press, pp. 439 – 446, August 1985.

Zacharias, F., Borst, C., and Hirzinger, G., "Online Generation of Reachable Grasps for Dexterous Manipulation Using a Representation of the Reachable Workspace," *International Conference on Advanced Robotics*, Munich, Germany, 2009a.

Zacharias, F., Sepp, W., Borst, C., and Hirzinger, G., "Using a Model of the Reachable Workspace to Position Mobile Manipulators for 3-D Trajectories," *9th IEEE-RAS International Conference on Humanoid Robots*, Paris, France, December 2009b.

Zacharias, F., Borst, C., Beetz, M., and Hirzinger, G., "Positioning Mobile Manipulators to Perform Constrained Linear Trajectories," *Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nice, France, September 2008.

Zacharias, F., Borst, C., and Hirzinger, G., "Capturing Robot Workspace Structure: Representing Robot Capabilities," *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, October 2007.

Zacharias, F., Borst, C., and Hirzinger, G., "Bridging the Gap between Task Planning and Path Planning," *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, October 2006.

Vita

Joshua Murry Williams was born in Phoenix, AZ, in 1985. He grew up in Humboldt, AZ, and attended Bradshaw Mountain High School, where he graduated as Co-Valedictorian in 2004. Joshua studied physics and played soccer at Occidental College in Los Angeles, graduating Magna Cum Laude in 2008. He entered the Mechanical Engineering program at The University of Texas at Austin to study nuclear engineering and robotics. While in graduate school, Joshua has completed two summer internships at Los Alamos National Lab. Joshua currently resides in Austin, TX.

Email: humboldtian@gmail.com

This thesis was typed by the author.